Timed Parity Games: Complexity and Robustness[*]

Krishnendu Chatterjee[1], Thomas A. Henzinger[2,3], and Vinayak S. Prabhu[2]

[1] CCE, UC Santa Cruz; [2] EECS, UC Berkeley; [3] CCS, EPFL;
{c_krish,vinayak}@eecs.berkeley.edu, tah@epfl.ch

Abstract. We consider two-player games played in real time on game structures with clocks and parity objectives. The games are *concurrent* in that at each turn, both players independently propose a time delay and an action, and the action with the shorter delay is chosen. To prevent a player from winning by blocking time, we restrict each player to strategies that ensure that the player cannot be responsible for causing a zeno run. First, we present an efficient reduction of these games to *turn-based* (i.e., nonconcurrent) *finite-state* (i.e., untimed) parity games. The states of the resulting game are pairs of clock regions of the original game. Our reduction improves the best known complexity for solving timed parity games. Moreover, the rich class of algorithms for classical parity games can now be applied to timed parity games.

Second, we consider two restricted classes of strategies for the player that represents the controller in a real-time synthesis problem, namely, *limit-robust* and *bounded-robust* strategies. Using a limit-robust strategy, the controller cannot choose an exact real-valued time delay but must allow for some nonzero jitter in each of its actions. If there is a given lower bound on the jitter, then the strategy is bounded-robust. We show that exact strategies are more powerful than limit-robust strategies, which are more powerful than bounded-robust strategies for any bound. For both kinds of robust strategies, we present efficient reductions to standard timed automaton games. These reductions provide algorithms for the synthesis of robust real-time controllers.

1 Introduction

Timed automata [3] are models of real-time systems in which states consist of discrete locations and values for real-time clocks. The transitions between locations are dependent on the clock values. *Timed automaton games* [9, 1, 7, 12, 11] are used to distinguish between the actions of several players (typically a "controller" and a "plant"). We consider two-player timed automaton games with ω-regular objectives specified as *parity conditions*. The class of ω-regular objectives can express all safety and liveness specifications that arise in the synthesis and verification of reactive systems, and parity conditions are a canonical form to express ω-regular objectives [20]. The construction of a winning strategy for player 1 in such games corresponds to the *controller-synthesis problem for real-time systems* [10, 17, 22] with respect to achieving a desired ω-regular objective.

Timed automaton games proceed in an infinite sequence of rounds. In each round, both players simultaneously propose moves, with each move consisting of an action and a time delay after which the player wants the proposed action to take place. Of the two proposed moves, the move with the shorter time delay "wins" the round and determines the next state of the game. Let a set Φ of runs be the desired objective for player 1. Then player 1 has a winning strategy for Φ if she has a strategy to ensure that, no matter what player 2 does, one of the following two conditions hold: (1) time diverges and the resulting run belongs to Φ, or (2) time does not diverge but player-1's moves are chosen only finitely often (and thus she is not to be blamed for the convergence of time) [9, 14].
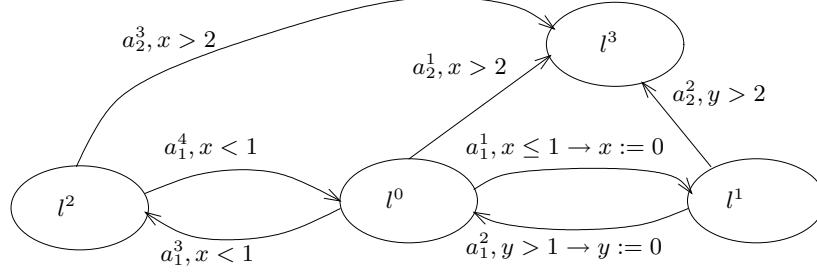
Fig. 1. A timed automaton game $\mathcal{T}$.

This definition of winning is equivalent to restricting both players to play according to *receptive* strategies [4, 19], which do not allow a player to win by blocking time.

In timed automaton games, there are cases where a player can win by proposing a certain strategy of moves, but where moves that deviate in the timing by an arbitrarily small amount from the winning strategy moves lead to her losing. If this is the case, then the synthesized controller needs to work with infinite precision in order to achieve the control objective. As this requirement is unrealistic, we propose two notions of *robust winning strategies*. In the first robust model, each move of player 1 (the "controller") must allow some jitter in when the action of the move is taken. The jitter may be arbitrarily small, but it must be greater than 0. We call such strategies *limit-robust*. In the second robust model, we give a lower bound on the jitter, i.e., every move of player 1 must allow for a fixed jitter, which is specified as a parameter for the game. We call these strategies *bounded-robust*. The strategies of player 2 (the "plant") are left unrestricted (apart from being receptive). We show that these types of strategies are in strict decreasing order in terms of power: general strategies are strictly more powerful than limit-robust strategies; and limit-robust strategies are strictly more powerful than bounded-robust strategies for *any* lower bound on the jitter, i.e., there are games in which player 1 can win with a limit-robust strategy, but there does not exist any nonzero bound on the jitter for which player 1 can win with a bounded-robust strategy. The following example illustrates this issue.

Example 1. Consider the timed automaton $\mathcal{T}$ in Fig. 1. The edges denoted a_1^k for $k \in \{1, 2, 3, 4\}$ are controlled by player 1 and edges denoted a_2^j for $j \in \{1, 2, 3\}$ are controlled by player 2. The objective of player 1 is $\Box(\neg l^3)$, ie., to avoid l^3. The important part of the automaton is the cycle l^0, l^1. The only way to avoid l^3 in a time divergent run is to cycle in between l^0 and l^1 infinitely often. In addition player 1 may choose to also cycle in between l^0 and l^2, but that does not help (or harm) her. Due to strategies being receptive, player 1 cannot just cycle in between l^0 and l^2 forever, she must also cycle in between l^0 and l^1; that is, to satisfy $\Box(\neg l^3)$ player 1 must ensure $(\Box\Diamond l^0) \wedge (\Box\Diamond l^1)$, where $\Box\Diamond$ denotes "infinitely often". But note that player 1 may cycle in between l^0 and l^2 as many (finite) number of times as she wants in between an l^0, l^1 cycle.

In our analysis below, we omit such l^0, l^2 cycles for simplicity. Let the game start from the location l^0 at time 0, and let l^1 be visited at time t^0 for the first time. Also, let t^j denote the difference between times when l^1 is visited for the j-th time, and when l^0 is visited for the j-th time. We can have at most 1 time unit between two successive visits to l^0, and we must have strictly more than 1 time unit elapse between two successive visits to l^1. Thus, t^j must be in a strictly decreasing sequence. Also, for player 1 to cycle around l^0 and l^1 infinitely often, we must have that all $t^j \geq 0$. Consider any bounded-robust strategy. Since the jitter is some fixed ε_j, for any strategy

of player 1 which tries to cycle in between l^0 and l^1, there will be executions where the transition labeled a_1^1 will be taken when x is less than or equal to $1 - \varepsilon_{\mathsf{j}}$, and the transition labeled a_1^2 will be taken when y is greater than $1 - \varepsilon_{\mathsf{j}}$. This means that there are executions where t^j decreases by at least $2 \cdot \varepsilon_{\mathsf{j}}$ in each cycle. But, this implies that we cannot having an infinite decreasing sequence of t^j's for any ε_{j} and for any starting value of t^0.

With a limit-robust strategy however, player 1 can cycle in between the two locations infinitely often, provided that the starting value of x is strictly less than 1. This is because at each step of the game, player 1 can pick moves that are such that the clocks x and y are closer and closer to 1 respectively. A general strategy allows player 1 to win even when the starting value of x is 1. The details will be presented later in Example 3 in subsection 4.2. □

Contributions. We first show that timed automaton parity games can be reduced to classical *turn-based* finite-state parity games. Since the timed games are *concurrent*, in that in each turn both players propose moves before one of the moves is chosen, our reduction to the untimed turn based game generates states that are pairs of clock regions. The reduction allows us to use the rich literature of algorithms for classical parity games to solve timed automaton parity games. While a solution for timed automaton games with parity objectives was already presented in [9], our reduction obtains a better computational complexity; we improve the complexity from roughly $O\left((M \cdot |C| \cdot |A_1| \cdot |A_2|)^2 \cdot (16 \cdot |S_{\mathsf{Reg}}|)^{d+2}\right)$ to roughly $O\left(M \cdot |C| \cdot |A_2|^* \cdot (32 \cdot |S_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|^*)^{\frac{d+2}{3} + \frac{3}{2}}\right)$, where M is the maximum constant in the timed automaton, $|C|$ is the number of clocks, $|A_i|$ is the number of player-i edges, $|A_i|^* = \min\{|A_i|, |L| \cdot 2^{|C|}\}$, $|L|$ is the number of of locations, $|S_{\mathsf{Reg}}|$ is the number of states in the region graph (bounded by $|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C|! \cdot 2^{|C|}$) , and d is the number of priorities in the parity index function. We note that the restriction to receptive strategies does not fundamentally change the complexity —it only increases the number of indices of the parity function by 2.

Second, we show that timed automaton games with limit-robust and bounded-robust strategies can be solved by reductions to general timed automaton games (with exact strategies). The reductions differentiate between whether the jitter is controlled by player 1 (in the limit-robust case), or by player 2 (in the bounded robust case). This is done by changing the winning condition in the limit-robust case, and by a syntactic transformation in the bounded-robust case. These reductions provide algorithms for synthesizing robust controllers for real-time systems, where the controller is guaranteed to achieve the control objective even if its time delays are subject to jitter. We also demonstrate that limit-robust strategies suffice for winning the special case of timed automaton games where all guards and invariants are strict (i.e., open). The question of the *existence* of a lower bound on the jitter for which a game can be won with a bounded-robust strategy remains open.

Related work. A solution for timed automaton games with receptive strategies and parity objectives was first presented in [9], where the solution is obtained by first demonstrating that the winning set can be characterized by a μ-calculus fixpoint expression, and then showing that only unions of clock regions arise in its fixpoint iteration. Our notion of bounded-robustness is closely related to the Almost-ASAP semantics of [24]. The work there is done in a one-player setting where the controller is already known, and one wants the know if the composition of the controller and the system satisfies a safety property in the presence of bounded jitter and observation delay. A similar model for hybrid automata is considered in [2]. The solution for the existence of bounded

jitter and observation delay for which a timed system stays safe is presented in [23]. Various models of robust timed automata (the one-player case) are also considered in [5, 6, 13, 15].

2   Timed Games

In this section we present the definitions of timed game structures, runs, objectives, and strategies in timed game structures.

Timed game structures. A *timed game structure* is a tuple $\mathcal{G} = \langle S, A_1, A_2, \Gamma_1, \Gamma_2, \delta \rangle$ with the following components.

- S is a set of states.
- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively. We assume that $\perp_i \notin A_i$, and write $A_i^{\perp}$ for $A_i \cup \{\perp_i\}$. The set of *moves* for player i is $M_i = \mathbb{R}_{\geq 0} \times A_i^{\perp_i}$. Intuitively, a move $\langle \Delta, a_i \rangle$ by player i indicates a waiting period of Δ time units followed by a discrete transition labeled with action a_i.
- $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ are two move assignments. At every state s, the set $\Gamma_i(s)$ contains the moves that are available to player i. We require that $\langle 0, \perp_i \rangle \in \Gamma_i(s)$ for all states $s \in S$ and $i \in \{1, 2\}$. Intuitively, $\langle 0, \perp_i \rangle$ is a time-blocking stutter move.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is the transition function. We require that for all time delays $\Delta, \Delta' \in \mathbb{R}_{\geq 0}$ with $\Delta' \leq \Delta$, and all actions $a_i \in A_i^{\perp_i}$, we have (1) $\langle \Delta, a_i \rangle \in \Gamma_i(s)$ iff both $\langle \Delta', \perp_i \rangle \in \Gamma_i(s)$ and $\langle \Delta - \Delta', a_i \rangle \in \Gamma_i(\delta(s, \langle \Delta', \perp_i \rangle))$; and (2) if $\delta(s, \langle \Delta', \perp_i \rangle) = s'$ and $\delta(s', \langle \Delta - \Delta', a_i \rangle) = s''$, then $\delta(s, \langle \Delta, a_i \rangle) = s''$.

The game proceeds as follows. If the current state of the game is s, then both players simultaneously propose moves $\langle \Delta_1, a_1 \rangle \in \Gamma_1(s)$ and $\langle \Delta_2, a_2 \rangle \in \Gamma_2(s)$. If $a_1 \neq \perp_1$, the move with the shorter duration "wins" in determining the next state of the game. If both moves have the same duration, then the next state is chosen non-deterministically. If $a_1 = \perp_1$, then the move of player 2 determines the next state, regardless of Δ_i. We give this special power to player 1 as the controller always has the option of letting the state evolve in a controller-plant framework, without always having to provide inputs to the plant. Formally, we define the *joint destination function* $\delta_{\mathsf{jd}} : S \times M_1 \times M_2 \mapsto 2^S$ by

$$
\delta_{\mathsf{jd}}(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle) = \begin{cases} \{\delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_1 < \Delta_2 \text{ and } a_1 \neq \perp_1; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle)\} & \text{if } \Delta_2 < \Delta_1 \text{ or } a_1 = \perp_1; \\ \{\delta(s, \langle \Delta_2, a_2 \rangle), \delta(s, \langle \Delta_1, a_1 \rangle)\} & \text{if } \Delta_2 = \Delta_1 \text{ and } a_1 \neq \perp_1. \end{cases}
$$

The time elapsed when the moves $m_1 = \langle \Delta_1, a_1 \rangle$ and $m_2 = \langle \Delta_2, a_2 \rangle$ are proposed is given by $\mathsf{delay}(m_1, m_2) = \min(\Delta_1, \Delta_2)$. The boolean predicate $\mathsf{blame}_i(s, m_1, m_2, s')$ indicates whether player i is "responsible" for the state change from s to s' when the moves m_1 and m_2 are proposed. Denoting the opponent of player i by $\sim i = 3 - i$, for $i \in \{1, 2\}$, we define

$$
\mathsf{blame}_i(s, \langle \Delta_1, a_1 \rangle, \langle \Delta_2, a_2 \rangle, s') = \left(\Delta_i \leq \Delta_{\sim i} \ \wedge \ \delta(s, \langle \Delta_i, a_i \rangle) = s' \right) \wedge (i = 1 \to a_1 \neq \perp_1).
$$

Runs. A *run* of the timed game structure $\mathcal{G}$ is an infinite sequence $r = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ such that $s_k \in S$ and $m_i^k \in \Gamma_i(s_k)$ and $s_{k+1} \in \delta_{\mathsf{jd}}(s_k, m_1^k, m_2^k)$ for all $k \geq 0$ and $i \in \{1, 2\}$. For $k \geq 0$, let $\mathsf{time}(r, k)$ denote the "time" at position k of the run, namely, $\mathsf{time}(r, k) = \sum_{j=0}^{k-1} \mathsf{delay}(m_1^j, m_2^j)$ (we let $\mathsf{time}(r, 0) = 0$). By $r[k]$ we denote the $(k+1)$-th state s_k of r. The run prefix $r[0..k]$ is the

finite prefix of the run r that ends in the state s_k. Let Runs be the set of all runs of $\mathcal{G}$, and let FinRuns be the set of run prefixes.

Objectives. An *objective* for the timed game structure $\mathcal{G}$ is a set $\Phi \subseteq$ Runs of runs. We will be interested in parity objectives. Parity objectives are canonical forms for ω-regular properties that can express all commonly used specifications that arise in verification.

Let $\Omega : S \mapsto \{0, \ldots, k-1\}$ be a parity index function. The parity objective for Ω requires that the maximal index visited infinitely often is even. Formally, let $\mathsf{InfOften}(\Omega(r))$ denote the set of indices visited infinitely often along a run r. Then the parity objective defines the following set of runs: $\mathsf{Parity}(\Omega) = \{r \mid \max(\mathsf{InfOften}(\Omega(r)))$ is even $\}$. A timed game structure $\mathcal{G}$ together with the index function Ω constitute a *parity timed game* (of *order k*) in which the objective of player 1 is $\mathsf{Parity}(\Omega)$.

Strategies. A *strategy* for a player is a recipe that specifies how to extend a run. Formally, a *strategy* π_i for player $i \in \{1, 2\}$ is a function π_i that assigns to every run prefix $r[0..k]$ a move m_i in the set of moves available to player i at the state $r[k]$. For $i \in \{1, 2\}$, let Π_i be the set of strategies for player i. Given two strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, the set of possible *outcomes* of the game starting from a state $s \in S$ is the set of possible runs denoted by $\mathsf{Outcomes}(s, \pi_1, \pi_2)$.

Receptive strategies. We will be interested in strategies that are meaningful (in the sense that they do not block time). To define them formally we first present the following two sets of runs.

- A run r is *time-divergent* if $\lim_{k \to \infty} \mathsf{time}(r, k) = \infty$. We denote by Timediv is the set of all time-divergent runs.
- The set $\mathsf{Blameless}_i \subseteq$ Runs consists of the set of runs in which player i is responsible only for finitely many transitions. A run $s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots$ belongs to the set $\mathsf{Blameless}_i$, for $i = \{1, 2\}$, if there exists a $k \geq 0$ such that for all $j \geq k$, we have $\neg\, \mathsf{blame}_i(s_j, m_1^j, m_2^j, s_{j+1})$.

A strategy π_i is *receptive* if for all strategies $\pi_{\sim i}$, all states $s \in S$, and all runs $r \in \mathsf{Outcomes}(s, \pi_1, \pi_2)$, either $r \in$ Timediv or $r \in \mathsf{Blameless}_i$. Thus, no what matter what the opponent does, a receptive strategy of player i cannot be responsible for blocking time. Strategies that are not receptive are not physically meaningful. A timed game structure $\mathcal{G}$ is *well-formed* if both players have receptive strategies. We restrict our attention to well-formed timed game structures. We denote Π_i^R to be the set of receptive strategies for player i. Note that for $\pi_1 \in \Pi_1^R, \pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq$ Timediv.

Winning sets. Given an objective Φ, let $\mathsf{WinTimeDiv}_1^{\mathcal{G}}(\Phi)$ denote the set of states s in $\mathcal{G}$ such that player 1 has a receptive strategy $\pi_1 \in \Pi_1^R$ such that for all receptive strategies $\pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \Phi$. The strategy π is said to be winning strategy. In computing the winning sets, we shall quantify over *all* strategies, but modify the objective to take care of time divergence. Given an objective Φ, let $\mathsf{TimeDivBl}_1(\Phi) = (\mathsf{Timediv} \cap \Phi) \cup (\mathsf{Blameless}_1 \setminus \mathsf{Timediv})$, i.e., $\mathsf{TimeDivBl}_1(\Phi)$ denotes the set of runs such that either time diverges and Φ holds, or else time converges and player 1 is not responsible for time to converge. Let $\mathsf{Win}_1^{\mathcal{G}}(\Phi)$ be the set of states in $\mathcal{G}$ such that for all $s \in \mathsf{Win}_1^{\mathcal{G}}(\Phi)$, player 1 has a (possibly non-receptive) strategy $\pi_1 \in \Pi_1$ such that for all (possibly non-receptive) strategies $\pi_2 \in \Pi_2$, we have $\mathsf{Outcomes}(s, \pi_1, \pi_2) \subseteq \Phi$. The strategy π_1 is said to be winning for the non-receptive game. The following result establishes the connection between Win and WinTimeDiv sets.

Theorem 1 ([14]). *For all well-formed timed game structures $\mathcal{G}$, and for all ω-regular objectives Φ, we have $\mathsf{Win}_1^{\mathcal{G}}(\mathsf{TimeDivBl}_1(\Phi)) = \mathsf{WinTimeDiv}_1^{\mathcal{G}}(\Phi)$.*

We now define a special class of timed game structures, namely, timed automaton games.

Timed automaton games. Timed automata [3] suggest a finite syntax for specifying infinite-state timed game structures. A *timed automaton game* is a tuple $\mathcal{T} = \langle L, C, A_1, A_2, E, \gamma \rangle$ with the following components:

- L is a finite set of locations.
- C is a finite set of clocks.
- A_1 and A_2 are two disjoint sets of actions for players 1 and 2, respectively.
- $E \subseteq L \times (A_1 \cup A_2) \times \mathsf{Constr}(C) \times L \times 2^C$ is the edge relation, where the set $\mathsf{Constr}(C)$ of *clock constraints* is generated by the grammar

$$\theta ::= x \leq d \mid d \leq x \mid \neg\theta \mid \theta_1 \wedge \theta_2$$

  for clock variables $x \in C$ and nonnegative integer constants d. For an edge $e = \langle l, a_i, \theta, l', \lambda \rangle$, the clock constraint θ acts as a guard on the clock values which specifies when the edge e can be taken, and by taking the edge e, the clocks in the set $\lambda \subseteq C$ are reset to 0. We require that for all edges $\langle l, a_i, \theta', l', \lambda' \rangle \neq \langle l, a_i', \theta'', l'', \lambda'' \rangle \in E$, we have $a_i \neq a_i'$. This requirement ensures that a state and a move together uniquely determine a successor state.
- $\gamma : L \mapsto \mathsf{Constr}(C)$ is a function that assigns to every location an invariant for both players. All clocks increase uniformly at the same rate. When at location l, each player i must propose a move out of l before the invariant $\gamma(l)$ expires. Thus, the game can stay at a location only as long as the invariant is satisfied by the clock values.

A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$ that maps every clock to a nonnegative real. The set of all clock valuations for C is denoted by $K(C)$. Given a clock valuation $\kappa \in K(C)$ and a time delay $\Delta \in \mathbb{R}_{\geq 0}$, we write $\kappa + \Delta$ for the clock valuation in $K(C)$ defined by $(\kappa + \Delta)(x) = \kappa(x) + \Delta$ for all clocks $x \in C$. For a subset $\lambda \subseteq C$ of the clocks, we write $\kappa[\lambda := 0]$ for the clock valuation in $K(C)$ defined by $(\kappa[\lambda := 0])(x) = 0$ if $x \in \lambda$, and $(\kappa[\lambda := 0])(x) = \kappa(x)$ if $x \notin \lambda$. A clock valuation $\kappa \in K(C)$ *satisfies* the clock constraint $\theta \in \mathsf{Constr}(C)$, written $\kappa \models \theta$, if the condition θ holds when all clocks in C take on the values specified by κ. A *state* $s = \langle l, \kappa \rangle$ of the timed automaton game $\mathcal{T}$ is a location $l \in L$ together with a clock valuation $\kappa \in K(C)$ such that the invariant at the location is satisfied, that is, $\kappa \models \gamma(l)$. We let S be the set of all states of $\mathcal{T}$. Given a timed automaton game $\mathcal{T}$, the definition of a associated timed game structure $[\![\mathcal{T}]\!]$ is standard [9].

Clock regions. Timed automaton games can be solved using a region construction from the theory of timed automata [3]. For a real $t \geq 0$, let $\mathsf{frac}(t) = t - \lfloor t \rfloor$ denote the fractional part of t. Given a timed automaton game $\mathcal{T}$, for each clock $x \in C$, let c_x denote the largest integer constant that appears in any clock constraint involving x in $\mathcal{T}$ (let $c_x = 1$ if there is no clock constraint involving x). Two states $\langle l_1, \kappa_1 \rangle$ and $\langle l_1, \kappa_1 \rangle$ are said to be *region equivalent* if all the following conditions are satisfied: (a) $l_1 = l_2$, (b) for all clocks x, $\kappa_1(x) \leq c_x$ iff $\kappa_2(x) \leq c_x$, (c) for all clocks x with $\kappa_1(x) \leq c_x$, $\lfloor \kappa_1(x) \rfloor = \lfloor \kappa_2(x) \rfloor$, (d) for all clocks x, y with $\kappa_1(x) \leq c_x$ and $\kappa_1(y) \leq c_y$, $\mathsf{frac}(\kappa_1(x)) \leq \mathsf{frac}(\kappa_1(x))$ iff $\mathsf{frac}(\kappa_2(x)) \leq \mathsf{frac}(\kappa_2(x))$, and (e) for all clocks x with $\kappa_1(x) \leq c_x$, $\mathsf{frac}(\kappa_1(x)) = 0$ iff $\mathsf{frac}(\kappa_2(x)) = 0$. A *region* is an equivalence class of states with respect to the region equivalence relation. There are finitely many clock regions; more precisely, the number of clock regions is bounded by $|L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C|! \cdot 2^{|C|}$.

Region strategies and objectives. For a state $s \in S$, we write $\mathsf{Reg}(s) \subseteq S$ for the clock region containing s. For a run r, we let the *region sequence* $\mathsf{Reg}(r) = \mathsf{Reg}(r[0]), \mathsf{Reg}(r[1]), \cdots$. Two runs

r, r' are region equivalent if their region sequences are the same. An ω-regular objective Φ is a region objective if for all region-equivalent runs r, r', we have $r \in \Phi$ iff $r' \in \Phi$. A strategy π_1 is a *region strategy*, if for all runs r_1 and r_2 and all $k \geq 0$ such that $\mathsf{Reg}(r_1[0..k]) = \mathsf{Reg}(r_2[0..k])$, we have that if $\pi_1(r_1[0..k]) = \langle \Delta, a_1 \rangle$, then $\pi_1(r_2[0..k]) = \langle \Delta', a_1 \rangle$ with $\mathsf{Reg}(r_1[k] + \Delta) = \mathsf{Reg}(r_2[k] + \Delta')$. The definition for player 2 strategies is analogous. Two region strategies π_1 and π_1' are region-equivalent if for all runs r and all $k \geq 0$ we have that if $\pi_1(r[0..k]) = \langle \Delta, a_1 \rangle$, then $\pi_1'(r[0..k]) = \langle \Delta', a_1 \rangle$ with $\mathsf{Reg}(r[k] + \Delta) = \mathsf{Reg}(r[k] + \Delta')$. A parity index function Ω is a region (resp. location) parity index function if $\Omega(s_1) = \Omega(s_2)$ whenever $\mathsf{Reg}(s_1) = \mathsf{Reg}(s_2)$ (resp. s_1, s_2 have the same location). Henceforth, we shall restrict our attention to region and location objectives.

Encoding time-divergence by enlarging the game structure. Given a timed automaton game $\mathcal{T}$, consider the enlarged game structure $\widehat{\mathcal{T}}$ with the state space $\widehat{S} \subseteq S \times \mathbb{R}_{[0,1)} \times \{\text{TRUE}, \text{FALSE}\}^2$, and an augmented transition relation $\widehat{\delta} : \widehat{S} \times (M_1 \cup M_2) \mapsto \widehat{S}$. In an augmented state $\langle s, \mathfrak{z}, tick, bl_1 \rangle \in \widehat{S}$, the component $s \in S$ is a state of the original game structure $[\![\mathcal{T}]\!]$, $\mathfrak{z}$ is value of a fictitious clock z which gets reset to 0 every time it hits 1, $tick$ is true iff z hit 1 at last transition and bl_1 is true if player 1 is to blame for the last transition. Note that any strategy π_i in $[\![\mathcal{T}]\!]$, can be considered a strategy in $\widehat{\mathcal{T}}$. The values of the clock z, $tick$ and bl_1 correspond to the values each player keeps in memory in constructing his strategy. Any run r in $\mathcal{T}$ has a corresponding unique run $\widehat{r}$ in $\widehat{\mathcal{T}}$ with $\widehat{r}[0] = \langle r[0], 0, \text{FALSE}, \text{FALSE} \rangle$ such that r is a projection of $\widehat{r}$ onto $\mathcal{T}$. For an objective Φ, we can now encode time-divergence as: $\mathsf{TimeDivBl}_1(\Phi) = (\Box \Diamond\, tick \rightarrow \Phi) \wedge (\neg\Box\Diamond\, tick \rightarrow \Diamond\Box \neg\, bl_1)$. Let $\widehat{\kappa}$ be a valuation for the clocks in $\widehat{C} = C \cup \{z\}$. A state of $\widehat{\mathcal{T}}$ can then be considered as $\langle \langle l, \widehat{\kappa} \rangle, tick, bl_1 \rangle$. We extend the clock equivalence relation to these expanded states: $\langle \langle l, \widehat{\kappa} \rangle\, tick, bl_1 \rangle \cong \langle \langle l', \widehat{\kappa}' \rangle, tick', bl_1' \rangle$ iff $l = l', tick = tick', bl_1 = bl_1'$ and $\widehat{\kappa} \cong \widehat{\kappa}'$. Given a location l, and a set $\lambda \subseteq \widehat{C}$, we let $\widehat{R}[\text{loc} := l, \lambda := 0]$ denote the region $\{\langle l, \widehat{\kappa} \rangle \in \widehat{S} \mid$ there exist l' and $\widehat{\kappa}'$ with $\langle l', \widehat{\kappa}' \rangle \in \widehat{R}$ and $\widehat{\kappa}(x) = 0$ if $x \in \lambda, \widehat{\kappa}(x) = \widehat{\kappa}'(x)$ if $x \notin \lambda\}$. For every ω-regular region objective Φ of $\mathcal{T}$, we have $\mathsf{TimeDivBl}_1(\Phi)$ to be an ω-regular region objective of $\widehat{\mathcal{T}}$.

We now present a lemma that states for region ω-regular objectives region winning strategies exist, and all strategies region-equivalent to a region winning strategy are also winning.

Lemma 1 ([8]). *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. Then, (1) there exists a region winning strategy for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$, and (2) if π_1' is a strategy that is region-equivalent to a region winning strategy π_1, then π_1' is a winning strategy for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

3 Exact Winning of Timed Parity Games

In this section we shall present a reduction of timed automaton games to turn-based finite game graphs. The reduction allows us to use the rich literature of algorithms for finite game graphs for solving timed automaton games. It also leads to algorithms with better complexity than the one presented in [9]. Let $\mathcal{T}$ be a timed automaton game, and let $\widehat{\mathcal{T}}$ be the corresponding enlarged timed game structure that encodes time divergence. We shall construct a finite state turn based game structure $\mathcal{T}^f$ based on regions of $\widehat{\mathcal{T}}$ which can be used to compute winning states for parity objectives for the timed automaton game $\mathcal{T}$. In this finite state game, first player 1 proposes a destination region $\widehat{R}_1$ together with a discrete action a_1. Intuitively, this can be taken to mean that in the game $\widehat{\mathcal{T}}$, player 1 wants to first let time elapse to get to the region $\widehat{R}_1$, and then take the discrete action a_1. Let us denote this intermediate state which specifies the desired region of player 1 in $\mathcal{T}^f$

by the tuple $\langle \widehat{R}, \widehat{R}_1, a_1 \rangle$. From this state in $\mathcal{T}^f$, player 2 similarly also proposes a move consisting of a region $\widehat{R}_2$ together with a discrete action a_2. These two moves signify that player i proposed a move $\langle \Delta_i, a_i \rangle$ in $\widehat{\mathcal{T}}$ from a state $\widehat{s} \in \widehat{R}$ such that $\widehat{s} + \Delta_i \in \widehat{R}_i$. The following lemma indicates that only the regions of $\widehat{s} + \Delta_i$ are important in determining the successor region in $\widehat{\mathcal{T}}$.

Lemma 2 ([8]). *Let $\mathcal{T}$ be a timed automaton game and let Y, Y_1', Y_2' be regions in the enlarged timed game structure $[\![\widehat{\mathcal{T}}]\!]$. Suppose player-i has a move from $s_1 \in Y$ to $s_1' \in Y'$, for $i \in \{1, 2\}$. Then, one of the following cases must hold.*

1. *From all states $\widehat{s} \in Y$, there exists a player-1 move $m_1^{\widehat{s}}$ with $\widehat{\delta}(\widehat{s}, m_1^{\widehat{s}}) \in Y_1'$ such that for all moves $m_2^{\widehat{s}}$ of player-2 with $\widehat{\delta}(\widehat{s}, m_2^{\widehat{s}}) \in Y_2'$, we have $\mathsf{blame}_1(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_1^{\widehat{s}})) = \text{TRUE}$ and $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_2^{\widehat{s}})) = \text{FALSE}$.*
2. *From all states $\widehat{s} \in Y$, for all moves $m_1^{\widehat{s}}$ of player-1 with $\widehat{\delta}(\widehat{s}, m_1^{\widehat{s}}) \in Y_1'$, there exists a player-2 move $m_2^{\widehat{s}}$ with $\widehat{\delta}(\widehat{s}, m_2^{\widehat{s}}) \in Y_2'$ such that $\mathsf{blame}_2(\widehat{s}, m_1^{\widehat{s}}, m_2^{\widehat{s}}, \widehat{\delta}(\widehat{s}, m_2^{\widehat{s}})) = \text{TRUE}$.*

By Lemma 2, given an initial state in $\widehat{R}$, for moves of both players to some fixed $\widehat{R}_1, \widehat{R}_2$, either the move of player 1 is always chosen, or player 2 can always pick a move such that player-1's move is foiled. Note that the lemma is asymmetric, the asymmetry arises in the case when time delays of the two moves result in the same region. In this case, not all moves of player 2 might work, but some will (e.g., a delay of player 2 that is the same as that for player 1).

Let $\widehat{S}_{\mathsf{Reg}} = \{x \mid X \text{ is a region of } \widehat{\mathcal{T}}\}$. Because of Lemma 2, we may construct a finite turn based game to capture the winning set. A *finite state turn based game* G consists of the tuple $\langle (S, E), (S_1, S_2) \rangle$, where (S_1, S_2) forms a partition of the finite set S of states, E is the set of edges, S_1 is the set of states from which only player 1 can make a move to choose an outgoing edge, and S_2 is the set of states from which only player 2 can make a move. The game is bipartite if every outgoing edge from a player-1 state leads to a player-2 state and vice-versa. A bipartite turn based finite game $\mathcal{T}^f = \langle (S^f, E^f), (\widehat{S}_{\mathsf{Reg}} \times \{1\}, \widehat{S}_{\mathsf{Tup}} \times \{2\}) \rangle$ can be constructed to capture the timed game $\mathcal{T}$ (the full construction can be found in the appendix). The state space S^f equals $\widehat{S}_{\mathsf{Reg}} \times \{1\} \cup \widehat{S}_{\mathsf{Tup}} \times \{2\}$. The set $\widehat{S}_{\mathsf{Reg}}$ is the set of regions of $\widehat{\mathcal{T}}$. Each $\langle \widehat{R}, 1 \rangle \in \widehat{S}_{\mathsf{Reg}} \times \{1\}$ is indicative of a state in the timed game $\widehat{\mathcal{T}}$ that belongs to the region $\widehat{R}$. Each $\langle Y, 2 \rangle \in \widehat{S}_{\mathsf{Tup}} \times \{2\}$ encodes the following information: (a) the previous state of $\mathcal{T}^f$ (which corresponds to a region $\widehat{R}$ of $\widehat{\mathcal{T}}$), (b) a region $\widehat{R}'$ of $\widehat{\mathcal{T}}$ (representing an intermediate state which results from time passage in $\widehat{\mathcal{T}}$ from a state in the previous region $\widehat{R}$ to a state in $\widehat{R}'$), and (c) the desired discrete action of player 1 to be taken from the intermediate state in $\widehat{R}'$. An edge from $\langle \widehat{R}, 1 \rangle$ to $\langle Y, 2 \rangle$ is represents the fact that in the timed game $\widehat{\mathcal{T}}$, from every state $\widehat{s} \in \widehat{R}$, player 1 has a move $\langle \Delta, a_1 \rangle$ such that $\widehat{s} + \Delta$ is in the intermediate region component $\widehat{R}'$ of $\langle Y, 2 \rangle$, with a_1 being the desired discrete action. From the state $\langle Y, 2 \rangle$, player 2 has moves to $\widehat{S}_{\mathsf{Reg}} \times \{1\}$ depending on what moves of player 2 in the timed game $\widehat{\mathcal{T}}$ can beat the player-1 moves from $\widehat{R}$ to $\widehat{R}'$ according to Lemma 2.

Each $Z \in S^f$ is itself a tuple, with the first component being a location of $\mathcal{T}$. Given a location parity index function Ω on $\mathcal{T}$, we let Ω^f be the parity index function on $\mathcal{T}^f$ such that $\Omega^f(\langle l, \cdot \rangle) = \Omega(\langle l, \cdot \rangle)$. Another parity index function $\widehat{\Omega}^f$ with two more priorities can be derived from Ω^f to take care of time divergence issues, as described in [9]. Given a set $X = X_1 \times \{1\} \cup X_2 \times \{2\} \subseteq S^f$, we let $\mathsf{RegStates}(X) = \{\widehat{s} \in \widehat{S} \mid \mathsf{Reg}(\widehat{s}) \in X_1\}$. Theorem 2 shows that the turn based game $\mathcal{T}^f$ captures the timed automaton game $\mathcal{T}$.

Theorem 2. *Let $\widehat{\mathfrak{T}}$ be an enlarged timed game structure, and let $\mathfrak{T}^f$ be the corresponding finite game structure. Then, given an ω-regular region objective $\mathsf{Parity}(\Omega)$, we have*
$$\mathsf{Win}_1^{\widehat{\mathfrak{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathfrak{T}^f}(\mathsf{Parity}(\widehat{\Omega}^f))).$$

Proof. A solution for obtaining the set $\mathsf{Win}_1^{\widehat{\mathfrak{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega)))$ has been presented in [9] using a μ-calculus formulation. The μ-calculus iteration uses the *controllable predecessor* operator for player 1, $\mathsf{CPre}_1 : 2^{\widehat{S}} \mapsto 2^{\widehat{S}}$, defined formally by $\widetilde{s} \in \mathsf{CPre}_1(Z)$ iff $\exists m_1 \in \widehat{\Gamma}_1(\widehat{s}) \; \forall m_2 \in \widehat{\Gamma}_2(\widehat{s}) \, . \, \delta_{\mathsf{jd}}(\widehat{s}, m_1, m_2) \subseteq Z$. Informally, $\mathsf{CPre}_1(Z)$ consists of the set of states from which player 1 can ensure that the next state will be in Z, no matter what player 2 does. It can be shown that CPre_1 preserves regions of $\widehat{\mathfrak{T}}$ using Lemma 2. We use the Pre_1 operator in turn based games: $\mathsf{Pre}_1(X) = \{s \in \widehat{S}_{\mathsf{Reg}} \times \{1\} \mid \exists s' \in X \text{ such that } (s, s') \in E^f\} \cup \{s \in \widehat{S}_{\mathsf{Tup}} \times \{2\} \mid \forall (s, s') \in E^f \text{ we have } s' \in X\}$. From the construction of $\mathfrak{T}^f$, it also follows that given $X = X_1 \times \{1\} \cup X_2 \times \{2\} \subseteq S^f$, we have

$$\mathsf{RegStates}(\mathsf{Pre}_1^{\mathfrak{T}^f}(\mathsf{Pre}_1^{\mathfrak{T}^f}(X))) = \mathsf{CPre}_1^{\widehat{\mathfrak{T}}}(\mathsf{RegStates}(X)) = \mathsf{CPre}_1^{\widehat{\mathfrak{T}}}(\mathsf{RegStates}(X_1)) \tag{1}$$

Let ϕ_c be the μ-calculus formula using the CPre_1 operator describing the winning set for $\mathsf{Parity}(\widehat{\Omega}) = \mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))$. Let ϕ_t be the μ-calculus formula using the Pre_1 operator in a turn based game describing the winning set for $\mathsf{Parity}(\widehat{\Omega})$. The formula ϕ_t can be obtained from ϕ_c by syntactically replacing every CPre_1 by Pre_1. Let the winning set for $\mathsf{Parity}(\widehat{\Omega})$ in $\mathfrak{T}^f$ be $W_1 \times \{1\} \cup W_2 \times \{2\}$. It is described by ϕ_t. The game in $\mathfrak{T}^f$ proceeds in a bipartite fashion — player 1 and player 2 alternate moves, with the state resulting from the move of player 1 having the same parity index as the originating state. Note that the objective $\mathsf{Parity}(\widehat{\Omega})$ depends only on the infinitely often occurring indices in the trace. Thus, $W_1 \times \{1\}$ can be also be described by the μ-calculus formula ϕ_t' obtained by replacing each Pre_1 in ϕ_t with $\mathsf{Pre}_1 \circ \mathsf{Pre}_1$, and taking states of the form $s \times \{1\}$ in the result. Since we are only interested in the set $W_1 \times \{1\}$, and since we have a bipartite game where the parity index remains the same for every next state of a player-1 state, the set $W_1 \times \{1\}$ can also be described by the μ-calculus formula ϕ_t'' obtained from ϕ_t' by intersecting every variable with $\widehat{S}_{\mathsf{Reg}} \times \{1\}$. Now, ϕ_t'' can be computed using a finite fixpoint iteration. Using the identity 1, we have that the sets in the fixpoint iteration computation of ϕ_t'' correspond to the sets in the fixpoint iteration computation of ϕ_c, that is, if $X \times \{1\}$ occurs in the computation of ϕ_t'' at stage j, then $\mathsf{RegStates}(X)$ occurs in the computation of ϕ_t'' at the same stage j. This implies that the sets are the same on termination for both ϕ_t'' and ϕ_c. Thus, $\mathsf{Win}_1^{\widehat{\mathfrak{T}}}(\mathsf{TimeDivBl}_1(\mathsf{Parity}(\Omega))) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathfrak{T}^f}(\mathsf{Parity}(\widehat{\Omega}^f)))$. $\qquad \square$

The state space of the finite turn based game can be seen to be at most $O(|\widehat{S}_{\mathsf{Reg}}|^2 \cdot |L| \cdot 2^{|C|})$ (a discrete action may switch the location, and reset some clocks). We show that it is not required to keep all possible pairs of regions, leading to a reduction in the size of the state space. This is because from a state $\widehat{s} \in R$, it is not possible to get all regions by letting time elapse.

Lemma 3. *Let $\mathfrak{T}$ be a timed automaton game, $\widehat{\mathfrak{T}}$ the corresponding enlarged game structure, and $\widehat{R}$ a region in $\widehat{\mathfrak{T}}$. The number of possible time successor regions of $\widehat{R}$ are at most $2 \cdot \sum_{x \in C} 2(c_x + 1) \leq 4 \cdot (M + 1) \cdot (|C| + 1)$, where c_x is the largest constant that clock x is compared to in $\widehat{\mathfrak{T}}$, $M = \max\{c_x \mid x \in C\}$ and C is the set of clocks in $\mathfrak{T}$.*

Complexity of reduction. Recall that for a timed automaton game $\mathfrak{T}$, A_i is the set of actions for player i, C is the set of clocks and M is the largest constant in $\mathfrak{T}$. Let $|A_i|^* = \min\{|A_i|, |L| \cdot 2^{|C|}\}$ and let $|\mathfrak{T}_{\mathsf{Constr}}|$ denote the length of the clock constraints in $\mathfrak{T}$. The size of the state space of $\mathfrak{T}^f$

is bounded by $|\widehat{S}_{\mathsf{Reg}}| \cdot (1 + (M + 1) \cdot (|C| + 2) \cdot 2 \cdot (|A_1|^* + 1))$, where $|\widehat{S}_{\mathsf{Reg}}| \leq 16 \cdot |L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C + 1|! \cdot 2^{|C|+1}$ is the number of regions of $\widehat{\mathcal{T}}$. The number of edges in $\mathcal{T}^f$ is bounded by $|\widehat{S}_{\mathsf{Reg}}| \cdot ((M + 1) \cdot (|C| + 2) \cdot 2) \cdot (|A_1|^* + 1) [(1 + (|A_2|^* + 1) \cdot ((M + 1) \cdot (|C| + 2) \cdot 2)]$. See the appendix for details.

Theorem 3. *Let $\mathcal{T}$ be a timed automaton game, and let Ω be a region parity index function of order d. The set $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega))$ can be computed in time*

$$O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + [M \cdot |C| \cdot |A_2|^*] \cdot \left[2 \cdot |\widehat{S}_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|^*\right]^{\frac{d+2}{3}+\frac{3}{2}}\right)$$

where $|\widehat{S}_{\mathsf{Reg}}| \leq 16 \cdot |L| \cdot \prod_{x \in C}(c_x + 1) \cdot |C + 1|! \cdot 2^{|C|+1}$, M is the largest constant in $\mathcal{T}$, $|\mathcal{T}_{\mathsf{Constr}}|$ is the length of the clock constraints in $\mathcal{T}$, C is the set of clocks, $|A_i|^* = \min\{|A_i|, |L| \cdot 2^{|C|}\}$, and $|A_i|$ the number of discrete actions of player i for $i \in \{1, 2\}$.

In Theorem 3, we have used the result from [18] which states that a turn based parity game with m edges, n states and d parity indices can be solved in $O(m \cdot n^{\frac{d}{3}+\frac{1}{2}})$ time. From Theorem 2, we can solve the finite state game $\mathcal{T}^f$ to compute winning sets for all ω-regular region parity objectives Φ for a timed automaton game $\mathcal{T}$, using *any* algorithm for finite state turn based games, e.g., strategy improvement, small-progress algorithms [21, 16]. Note that $\mathcal{T}^f$ does not depend on the parity condition used, and there is a correspondence between the regions repeating infinitely often in $\mathcal{T}$ and $\mathcal{T}^f$. Hence, it is not required to explicitly convert an ω-regular objective Φ to a parity objective to solve using the $\mathcal{T}^f$ construction. We can solve the finite state game $\mathcal{T}^f$ to compute winning sets for all ω-regular region objectives Φ, where Φ is a Muller objective. Since Muller objectives subsume Rabin, Streett (strong fairness objectives), parity objectives as a special case, our result holds for more a much richer class of objectives than parity objectives.

Corollary 1. *Let $\widehat{\mathcal{T}}$ be an enlarged timed game structure, and let $\mathcal{T}^f$ be the corresponding finite game structure. Then, given an ω-regular region objective Φ, where Φ is specified as a Muller objective, we have $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\mathsf{TimeDivBl}_1(\Phi)) = \mathsf{RegStates}(\mathsf{Win}_1^{\mathcal{T}^f}(\mathsf{TimeDivBl}_1(\Phi)))$.*

4 Robust Winning of Timed Parity Games

In this section we study restrictions on player-1 strategies to model robust winning, and show how the winning sets can be obtained by reductions to general timed automaton games. The results of Section 3 can then be used to obtain algorithms for computing the robust winning sets.

There is inherent uncertainty in real-time systems. In a physical system, an action may be prescribed by a controller, but the controller can never prescribe a single timepoint where that action will be taken with probability 1. There is usually some *jitter* when the specified action is taken, the jitter being non-deterministic. The model of general timed automaton games, where player 1 can specify exact moves of the form $\langle \Delta, a_1 \rangle$ consisting of an action together with a delay, assume that the jitter is 0. In subsection 4.1, we obtain robust winning sets for player 1 in the presence of non-zero jitter (which are assumed to be arbitrarily small) for each of her proposed moves. In subsection 4.2, we assume the the jitter to be some fixed $\varepsilon_j \geq 0$ for every move that is known. The strategies of player 2 are left unrestricted. In the case of lower-bounded jitter, we also introduce a *response time* for player-1 strategies. The response time is the minimum delay between

a discrete action, and a discrete action of the controller. We note that the set of player-1 strategies with a jitter of $\varepsilon_j > 0$ contains the set of player-1 strategies with a jitter of $\varepsilon_j/2$ and a response time of $\varepsilon_j/2$. Thus, the strategies of subsection 4.1 automatically have a response time greater than 0. The winning sets in both sections are hence robust towards the presence of jitter and response times.

4.1 Winning in the Presence of Jitter

In this subsection, we model games where the jitter is assumed to be greater than 0, but arbitrarily small in each round of the game.

Given a state s, a *limit-robust move* for player 1 is either the move $\langle \Delta, \bot_1 \rangle$ with $\langle \Delta, \bot_1 \rangle \in \Gamma_1(s)$; or it is a tuple $\langle [\alpha, \beta], a_1 \rangle$ for some $\alpha < \beta$ such that for every $\Delta \in [\alpha, \beta]$ we have $\langle \Delta, a_1 \rangle \in \Gamma_1(s)$. [1] Note that a time move $\langle \Delta, \bot_1 \rangle$ for player 1 implies that she is relinquishing the current round to player 2, as the move of player 2 will always be chosen, and hence we allow a singleton time move. Given a limit-robust move $mrob_1$ for player 1, and a move m_2 for player 2, the set of possible outcomes is the set $\{\delta_{jd}(s, m_1, m_2) \mid$ either (a) $mrob_1 = \langle \Delta, \bot_1 \rangle$ and $m_1 = mrob_1$; or (b) $mrob_1 = \langle [\alpha, \beta], a_1 \rangle$ and $m_1 = \langle \Delta, a_1 \rangle$ with $\Delta \in [\alpha, \beta]\}$. A *limit-robust strategy* π_1^{rob} for player 1 prescribes limit-robust moves to finite run prefixes. We let Π_1^{rob} denote the set of limit-robust strategies for player-1. Given an objective Φ, let $\mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$ denote the set of states s in $\mathcal{T}$ such that player 1 has a limit-robust receptive strategy $\pi_1^{\mathsf{rob}} \in \Pi_1^R$ such that for all receptive strategies $\pi_2 \in \Pi_2^R$, we have $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) \subseteq \Phi$. We say a limit-robust strategy π_1^{rob} is region equivalent to a strategy π_1 if for all runs r and for all $k \geq 0$, the following conditions hold: (a) if $\pi_1(r[0..k]) = \langle \Delta, \bot_1 \rangle$, then $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta', \bot_1 \rangle$ with $\mathsf{Reg}(r[k] + \Delta) = \mathsf{Reg}(r[k] + \Delta')$; and (b) if $\pi_1(r[0..k]) = \langle \Delta, a_1 \rangle$ with $a_1 \neq \bot_1$, then $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ with $\mathsf{Reg}(r[k] + \Delta) = \mathsf{Reg}(r[k] + \Delta')$ for all $\Delta' \in [\alpha, \beta]$. Note that for any limit-robust move $\langle [\alpha, \beta], a_1 \rangle$ with $a_1 \neq \bot_1$ from a state s, we must have that the set $\{s + \Delta \mid \Delta \in [\alpha, \beta]\}$ contains an open region of $\mathcal{T}$.

We now show how to compute the set $\mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$. Given a timed automaton game $\mathcal{T}$, we have the corresponding enlarged game structure $\widehat{\mathcal{T}}$ which encodes time-divergence. We add another boolean variable to $\widehat{\mathcal{T}}$ to obtain another game structure $\widehat{\mathcal{T}}_{\mathsf{rob}}$. The state space of $\widehat{\mathcal{T}}_{\mathsf{rob}}$ is $\widehat{S} \times \{\mathrm{TRUE}, \mathrm{FALSE}\}$. The transition relation $\widehat{\delta}_{\mathsf{rob}}$ is such that $\widehat{\delta}_{\mathsf{rob}}(\langle \widehat{s}, rb_1 \rangle, \langle \Delta, a_i \rangle) = \langle \widehat{\delta}(\widehat{s}, \langle \Delta, a_i \rangle), rb_1' \rangle$, where $rb_1' = \mathrm{TRUE}$ iff $rb_1 = \mathrm{TRUE}$ and one of the following hold: (a) $a_i \in A_2^\perp$; or (b) $a_i = \bot_1$; or (c) $a_i \in A_1$ and $s + \Delta$ belongs to an open region of $\widehat{\mathcal{T}}$.

Theorem 4. *Given a state s in a timed automaton game $\mathcal{T}$ and an ω-regular region objective Φ, we have $s \in \mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$ iff $\langle s, \cdot, \cdot, \cdot, \mathrm{TRUE} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}_{\mathsf{rob}}}(\Phi \wedge \Box(rb_1 = \mathrm{TRUE}) \wedge (\Diamond\Box(tick = \mathrm{FALSE}) \rightarrow (\Diamond\Box(bl_1 = \mathrm{FALSE}))))$.*

Proof. 1. ($\Rightarrow$) Suppose player-1 has a limit-robust receptive strategy winning strategy π_1 for Φ. starting from a state s in $\mathcal{T}$. we show $\langle s, \cdot, \cdot, \cdot, \mathrm{TRUE} \rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}_{\mathsf{rob}}}(\Phi \wedge \Box(rb_1 = \mathrm{TRUE}) \wedge (\Diamond\Box(tick = \mathrm{FALSE}) \rightarrow (\Diamond\Box(bl_1 = \mathrm{FALSE}))))$.

We may consider π_1 to be a strategy in $\widehat{\mathcal{T}}$. Since π_1 is a limit-robust strategy, player-1 proposes limit-robust moves at each step of the game. Given a state $\widehat{s}$, and a limit-robust move $\langle [\alpha, \beta], a_1 \rangle$, there always exists $\alpha < \alpha' < \beta' < \beta$ such that for every $\Delta \in [\alpha', \beta']$, we have that $\widehat{s} + \Delta$ belongs to an open region of $\widehat{\mathcal{T}}$. Thus, given any limit-robust strategy π_1, we can obtain another limit-robust strategy π_1' in $\widehat{\mathcal{T}}$, such that for every k, (a) if $\pi_1(r[k]) = \langle \Delta, \bot_1 \rangle$, then $\pi_1'(r[k]) = \pi_1(r[k])$;

---

[1] We can alternatively have an open, or semi-open time interval, the results do not change.

and (b) if $\pi_1(r[k]) = \langle[\alpha,\beta],a_1\rangle$, then $\pi_1'(r[k]) = \langle(\Delta,a_1\rangle$ with $\Delta \in [\alpha',\beta'] \subseteq [\alpha,\beta]$, and $\{r[k]+\Delta' \mid \Delta' \in [\alpha',\beta']\}$ being a subset of an open region of $\widehat{\mathcal{T}}$. Thus for any strategy π_2 of player-2, and for any run $r \in \mathsf{Outcomes}(\langle s,\cdot,\cdot,\mathrm{TRUE}\rangle,\pi_1',\pi_2)$, we have that r satisfies $\square(rb_1 = \mathrm{TRUE})$. Since π_1 was a receptive winning strategy for Φ, π_1' is also a receptive winning strategy for Φ. Hence, r also satisfies $\Phi \wedge \lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE})$.

2. ($\Leftarrow$) Suppose $\langle s,\cdot,\cdot,\mathrm{TRUE}\rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}\mathsf{rob}}(\Phi \wedge \square(rb_1 = \mathrm{TRUE}) \wedge (\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE}))))$. We show that player-1 has a limit-robust receptive winning strategy from state s. Let π_1 be a winning region winning strategy for player-1 for the objective $\Phi \wedge \square(rb_1 = \mathrm{TRUE}) \wedge (\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE})))$. For every run r starting from state $\langle s,\cdot,\cdot,\mathrm{TRUE}\rangle$, the strategy π_1 is such that $\pi_1(r[0..k]) = \langle\Delta^k,a_1^k\rangle$ such that either $a_1^k = \bot_1$, or $r[k] + \Delta^k$ belongs to an open region $\widehat{R}$ of $\widehat{S}$ Since R is an open region, there always exists some $\alpha < \beta$ such that for every $\Delta \in [\alpha,\beta]$, we have $r[k] + \Delta \in R$. Consider the strategy π_1^{rob} that prescribes a limit-robust move $\langle[\alpha,\beta],a_1^k\rangle$ for the history $r[0..k]$ if $\pi_1(r[0..k]) = \langle\Delta^k,a_1^k\rangle$ with $a_1^k \neq \bot_1$, and $\pi_1^{\mathsf{rob}}(r[0..k]) = \pi_1(r[0..k])$ otherwise. The strategy π_1^{rob} is region-equivalent to π_1, and hence is also winning for player-1 by a lemma similar to Lemma 1 (see Lemma 4 in the appendix). Since it only prescribes limit-robust moves, it is a limit-robust strategy. And since it ensures $\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE})$, it is a receptive strategy.

$\square$

We say a timed automaton $\mathcal{T}$ is *open* if all the guards and invariants in $\mathcal{T}$ are open. Note that even though all the guards and invariants are open, a player might still propose moves to closed regions, e.g., consider an edge between two locations l_1 and l_2 with the guard $0 < x < 2$; a player might propose a move from $\langle l_1, x = 0.2\rangle$ to $\langle l_2, x = 1\rangle$. The next theorem shows that this is not required of player 1 in general, that is, to win for an ω-regular location objective, player 1 only needs to propose moves to open regions of $\mathcal{T}$. Let $\mathsf{Constr}^*(C)$ be the set of clock constraints generated by the grammar

$$\theta ::= x < d \mid x > d \mid x \geq 0 \mid x < y \mid \theta_1 \wedge \theta_2$$

for clock variables $x, y \in C$ and nonnegative integer constants d. An *open polytope* of $\mathcal{T}$ is set of states X such that $X = \{\langle l,\kappa\rangle \in S \mid \kappa \models \theta\}$ for some $\theta \in \mathsf{Constr}^*(C)$. An open polytope X is hence a union of regions of $\mathcal{T}$. Note that it may contain open as well as closed regions. We say a parity objective $\mathsf{Parity}(\Omega)$ is an open polytope objective if $\Omega^{-1}(j)$ is an open polytope for every $j \geq 0$.

Theorem 5. *Let $\mathcal{T}$ be an open timed automaton game and let $\Phi = \mathsf{Parity}(\Omega)$ be an ω-regular location objective. Then, $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\Phi) = \mathsf{RobWinTimeDiv}_1^{\mathcal{T}}(\Phi)$.*

Proof. We present a sketch of the proof. We shall work on the expanded game structure $\widehat{\mathcal{T}}^{\mathsf{rob}}$, and prove that $\langle s,\cdot,\cdot,\mathrm{TRUE}\rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}\mathsf{rob}}(\Phi \wedge \square(rb_1 = \mathrm{TRUE}) \wedge (\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE}))))$ iff $\langle s,\cdot,\cdot,\mathrm{TRUE}\rangle \in \mathsf{Win}_1^{\widehat{\mathcal{T}}\mathsf{rob}}(\Phi \wedge (\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE}))))$. The desired result will then follow from Theorem 4.

Consider the objective $\mathsf{TimeDivBl}_1(\Phi) = \Phi \wedge (\lozenge\square(tick = \mathrm{FALSE}) \rightarrow (\lozenge\square(bl_1 = \mathrm{FALSE})))$. Let $\widehat{\Omega}$ be the parity index function such that $\mathsf{Parity}(\widehat{\Omega}) = \mathsf{TimeDivBl}_1(\Phi)$. Since Φ is a location objective, and all invariants are open, we have $\widehat{\Omega}^{-1}(j)$ to be an open polytope of $\widehat{\mathcal{T}}^{\mathsf{rob}}$ for all indices $j \geq 0$ (recall that a legal state of $\mathcal{T}$ must satisfy the invariant of the location it is in).

The winning set for a parity objective $\mathsf{Parity}(\widehat{\Omega})$ can be described by a μ-calculus formula, we illustrate the case for when $\widehat{\Omega}$ has only two priorities. The μ-calculus formula is

12

then: $\mu Y \nu X \left[(\widehat{\Omega}^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\widehat{\Omega}^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. This set can be computed from a (finite) iterative fixpoint procedure. Let $Y^* = \mu Y \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. The iterative fixpoint procedure computes $Y_0 = \emptyset \subseteq Y_1 \subseteq \cdots \subseteq Y_n = Y^*$, where $Y_{i+1} = \nu X \left[(\Omega^{-1}(1) \cap \mathsf{CPre}_1(Y_i)) \cup (\Omega^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. We claim that each Y_i for $i > 0$ is a union of open polytopes of $\widehat{\mathcal{T}}^{\mathsf{rob}}$. This is because (a) the union and intersection of a union of open polytopes is again a union of open polytopes, and (b) $\nu X (A \cup (B \cap \mathsf{CPre}_1(X)))$ is an open polytope provided A, B are open polytopes, and $\mathcal{T}$ is an open timed automaton game. We can consider the states in $Y_i \setminus Y_{i-1}$ as being added in two steps, T_{2i-1} and $T_{2i}(= Y_i)$ as follows:

1. $T_{2i-1} = \widehat{\Omega}^{-1}(1) \cap \mathsf{CPre}_1(Y_{i-1})$. T_{2i-1} is clearly a subset of Y_i.
2. $T_{2i} = \nu X \left[T_{2i-1} \cup (\widehat{\Omega}^{-1}(0) \cap \mathsf{CPre}_1(X)) \right]$. Note $(T_{2i} \setminus T_{2i-1}) \cap \widehat{\Omega}^{-1}(1) = \emptyset$.

Thus, in odd stages we add states with index 1, and in even stages we add states with index 0. The *rank* of a state $\widehat{s} \in Y^*$ is j if $\widehat{s} \in T_j \setminus \cup_{k=0}^{j-1} T_k$. Each rank thus consists of states forming an open polytope. A winning strategy for player 1 can also be obtained based on the fixpoint iteration. The requirements on a strategy to be a winning strategy based on the fixpoint schema are:

1. For a state of even rank j, the strategy for player 1 must ensure that she has a move such that against all moves of player 2, the next state either (a) has index 0 and belongs to the same rank or less, or (b) the next state has index 1 and belongs to rank smaller than j.
2. For a state of odd rank j, the strategy for player 1 must ensures that she has a move such that against all moves of player 2, the next state belongs to a lower rank.

Since the rank sets are all open polytopes, and $\mathcal{T}$ is an open timed automaton, we have that there exists a winning strategy which from every state in a region $\widehat{R}$, either proposes a pure time move, or proposes a move to an open region (as every open polytope must contain an open region). Hence, this particular winning strategy also ensures that $\square(rb_1 = \text{TRUE})$ holds. Thus, this strategy ensures $\mathsf{TimeDivBl}_1(\Phi) \wedge \square(rb_1 = \text{TRUE})$. The general case of an index function of order greater than two can be proved by an inductive argument. □

4.2 Winning with Bounded Jitter and Response Time

The limit-robust winning strategies described in subsection 4.1 did not have a lower bound on the jitter: player 1 could propose a move $\langle [\alpha, \alpha + \varepsilon], a_1 \rangle$ for arbitrarily small α and ε. In some cases, the controller may be required to work with a known jitter, and also a finite *response time*. Intuitively, the response time is the minimum delay between a discrete action and a discrete action of the controller. We model this scenario by allowing player 1 to propose moves with a single time point, but we make the jitter and the response time explicit and modify the semantics as follows. Player 1 can propose exact moves (with a delay greater than the response time), but the actual delay in the game will be controlled by player 2 and will be in a jitter interval around the proposed player-1 delay.

Given a finite run $r[0..k] = s_0, \langle m_1^0, m_2^0 \rangle, s_1, \langle m_1^1, m_2^1 \rangle, \ldots, s_k$, let $\mathsf{TimeElapse}(r[0..k]) = \sum_{j=p}^{k-1} \mathsf{delay}(m_1^j, m_2^j)$ where p is the least integer greater than or equal to 0 such that for all $k > j \geq p$ we have $m_2^j = \langle \Delta_2^j, \perp_2 \rangle$ and $\mathsf{blame}_2(s_j, m_1^j, m_2^j, s_{j+1}) = \text{TRUE}$ (we take $\mathsf{TimeElapse}(r[0..k]) = 0$ if $p = k$). Intuitively, $\mathsf{TimeElapse}(r[0..k])$ denotes the time that has passed due to a sequence of contiguous pure time moves leading upto s_k in the run $r[0..k]$. Let $\varepsilon_{\mathsf{j}} \geq 0$ and $\varepsilon_{\mathsf{r}} \geq 0$ be given

13

bounded jitter and response time (we assume both are rational). Since a pure time move of player 1 is a relinquishing move, we place no restriction on it. Player 2 can also propose moves such that only time advances, without any discrete action being taken. in this case, we need to adjust the remaining response time. Formally, an ε_{j}-*jitter* ε_{r}-*response bounded-robust strategy* π_1 of player 1 proposes a move $\pi_1(r[0..k]) = m_1^k$ such that either

- $m_1^k = \langle \Delta^k, \bot_1 \rangle$ with $\langle \Delta, \bot_1 \rangle \in \Gamma_1(S)$, or,
- $m_1^k = \langle \Delta^k, a_1 \rangle$ such that the following two conditions hold:
  - $\Delta^k \geq \max(0, \varepsilon_{\mathsf{r}} - \mathsf{TimeElapse}(r[0..k]))$, and,
  - $\langle \Delta', a_1 \rangle \in \Gamma_1(s)$ for all $\Delta' \in [\Delta^k, \Delta^k + \varepsilon_{\mathsf{j}}]$.

Given a move $m_1 = \langle \Delta, a_1 \rangle$ of player 1 and a move m_2 of player 2, the set of resulting states is given by $\delta_{\mathsf{jd}}(s, m_1, m_2)$ if $a_1 = \bot_1$, and by $\{\delta_{\mathsf{jd}}(s, m_1 + \epsilon, m_2) \mid \epsilon \in [0, \varepsilon_{\mathsf{j}}]\}$ otherwise. Given an ε_{j}-jitter ε_{r}-response bounded-robust strategy π_1 of player 1, and a strategy π_2 of player 2, the set of possible outcomes in the present semantics is denoted by $\mathsf{Outcomes}_{jr}(s, \pi_1, \pi_2)$. We denote the winning set for player 1 for an objective Φ given finite ε_{j} and ε_{r} by $\mathsf{JRWinTimeDiv}_1^{\mathcal{T}, \varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}(\Phi)$. We now show that $\mathsf{JRWinTimeDiv}_1^{\mathcal{T}, \varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}(\Phi)$ can be computed by obtaining a timed automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ from $\mathcal{T}$ such that $\mathsf{WinTimeDiv}_1^{\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}}(\Phi) = \mathsf{JRWinTimeDiv}_1^{\mathcal{T}, \varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}(\Phi)$.

Given a clock constraint φ we make the clocks appearing in φ explicit by denoting the constraint as $\varphi(\overrightarrow{x})$ for $\overrightarrow{x} = [x_1, \ldots, x_n]$. Given a real number δ, we let $\varphi(\overrightarrow{x} + \delta)$ denote the clock constraint φ' where φ' is obtained from φ by syntactically substituting $x_j + \delta$ for every occurrence of x_j in φ. Let $f^{\varepsilon_{\mathsf{j}}} : \mathsf{Constr}(C) \mapsto \mathsf{Constr}(C)$ be a function defined by $f^{\varepsilon_{\mathsf{j}}}(\varphi(\overrightarrow{x})) = \mathsf{ElimQuant}(\forall \delta \, (0 \leq \delta \leq \varepsilon_{\mathsf{j}} \rightarrow \varphi(\overrightarrow{x} + \delta)))$, where $\mathsf{ElimQuant}$ is a function that eliminates quantifiers (this function exists as we are working in the theory of reals with addition, which admits quantifier elimination). The formula $f^{\varepsilon_{\mathsf{j}}}(\varphi)$ ensures that φ holds at all the points in $\{\overrightarrow{x} + \Delta \mid \Delta \leq \varepsilon_{\mathsf{j}}\}$.

We now describe the timed automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$. The automaton has an extra clock z. The set of actions for player 1 is $\{\langle 1, e \rangle \mid e \text{ is a player-1 edge in } \mathcal{T}\}$ and for player 2 is $A_2 \cup \{\langle a_2, e \rangle \mid a_2 \in A_2 \text{ and } e \text{ is a player-1 edge in } \mathcal{T}\} \cup \{\langle 2, e \rangle \mid e \text{ is a player-1 edge in } \mathcal{T}\}$ (we assume the unions are disjoint). For each location l of $\mathcal{T}$ with the outgoing player-1 edges $e_1^1, \ldots, e_1^m$, the automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ has $m + 1$ locations: $l, l_{e_1^1}, \ldots, l_{e_1^m}$. Every edge of $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ includes z in its reset set. The invariant for l is the same as the invariant for l in $\mathcal{T}$. All player-2 edges of $\mathcal{T}$ are also player-2 edges in $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ (with the reset set being expanded to include z). The invariant for l_{e_j} is $z \leq \varepsilon_{\mathsf{j}}$. If $\langle l, a_2, \varphi, l', \lambda \rangle$ is an edge of $\mathcal{T}$ with $a_2 \in A_2$, then then $\langle l_{e_j}, \langle a_2, e_j \rangle, \varphi, l', \lambda \cup \{z\} \rangle$ is a player-2 edge of $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ for every player-1 edge e_j of $\mathcal{T}$. For every player-1 edge $e_j = \langle l, a_1^j, \varphi, l', \lambda \rangle$ of $\mathcal{T}$, the location l of $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ has the outgoing player-1 edge $\langle l, \langle 1, e_j \rangle, f^{\varepsilon_{\mathsf{j}}}\left(\gamma^{\mathcal{T}}(l)\right) \wedge (z \geq \varepsilon_{\mathsf{r}}) \wedge f^{\varepsilon_{\mathsf{j}}}(\varphi), l_{e_j}, \lambda \cup \{z\} \rangle$. The location l_{e_j} also has an additional outgoing *player-2* edge $\langle l_{e_j}, \langle 2, e_j \rangle, \varphi, l', \lambda \cup \{z\} \rangle$. The automaton $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ as described contains the rational constants ε_{r} and ε_{j}. We can change the timescale by multiplying every constant by the least common multiple of the denominators of ε_{r} and ε_{j} to get a timed automaton with only integer constants. Intuitively, in the game $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$, player 1 moving from l to l_{e_j} with the edge $\langle 1, e_j \rangle$ indicates the desire of player 1 to pick the edge e_j from location l in the game $\mathcal{T}$. This is possible in $\mathcal{T}$ iff the (a) more that ε_{r} time has passed since the last discrete action, (b) the edge e_j is enabled for at least ε_{j} more time units, and (c) the invariant of l is satisfied for at least ε_{j} more time units. These three requirements are captured by the new guard in $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$, namely $f^{\varepsilon_{\mathsf{j}}}\left(\gamma^{\mathcal{T}}(l)\right) \wedge (z \geq \varepsilon_{\mathsf{r}}) \wedge f^{\varepsilon_{\mathsf{j}}}(\varphi)$. The presence of jitter in $\mathcal{T}$ causes uncertainty in when exactly the edge e_j is taken. This is modeled in $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ by having the location l_{e_j} be controlled entirely by player 2 for a duration of ε_{j} time units. Within ε_{j} time units, player 2 must either propose a move $\langle a_2, e_j \rangle$
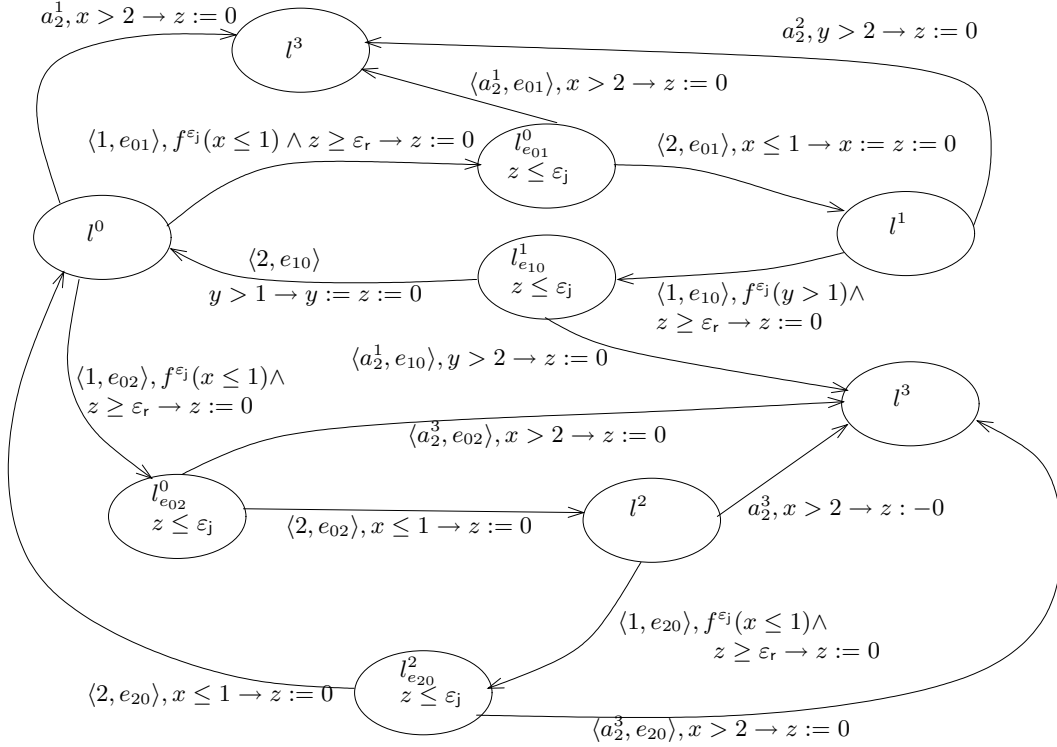
Fig. 2. The timed automaton game $\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}$ obtained from $\mathcal{T}$.

(corresponding to one of its own moves a_2 in $\mathcal{T}$, or allow the action $\langle 2, e_j \rangle$ (corresponding to the original player-1 edge e_j) to be taken. Given a parity function $\Omega^{\mathcal{T}}$ on $\mathcal{T}$, the parity function $\Omega^{\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}}$ is given by $\Omega^{\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}}(l) = \Omega^{\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}}(l_{e_j}) = \Omega^{\mathcal{T}}(l)$ for every player-1 edge e_j of $\mathcal{T}$. In computing the winning set for player 1, we need to modify blame_1 for technical reasons. Whenever an action of the form $\langle 1, e_j \rangle$ is taken, we blame player 2 (even though the action is controlled by player 1); and whenever an action of the form $\langle 2, e_j \rangle$ is taken, we blame player 1 (even though the action is controlled by player 2). Player 2 is blamed as usual for the actions $\langle a_2, e_j \rangle$. This modification is needed because player 1 taking the edge e_j in $\mathcal{T}$ is broken down into two stages in $\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}$. If player 1 to be blamed for the edge $\langle 1, e_j \rangle$, then the following could happen: (a) player 1 takes the edge $\langle 1, e_j \rangle$ in $\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}$ corresponding to her intention to take the edge e_j in $\mathcal{T}$ (b) player 2 then proposes her own move $\langle a_2, e_j \rangle$ from l_{e_j}, corresponding to her blocking the move e_j by a_2 in $\mathcal{T}$. If the preceeding scenario happens infinitely often, player 1 gets blamed infinitely often even though all she has done is signal her intentions infinitely often, but her actions have not been chosen. Hence player 2 is blamed for the edge $\langle 1, e_j \rangle$. If player 2 allows the intended player 1 edge by taking $\langle 2, e_j \rangle$, then we must blame player 1. We note that this modification is not required if $\varepsilon_r > 0$.

Example 2 (Construction of $\mathcal{T}^{\varepsilon_{j},\varepsilon_{r}}$). An example of the construction is given in Figure 2, corresponding to the timed automaton of Figure 1. The location l^3 is an absorbing location — it only has self-loops (we omit these self loops in the figures for simplicity). For the automaton $\mathcal{T}$, we have $A_1 = \{a_1^1, a_1^2, a_1^3, a_1^4\}$ and $A_2 = \{a_2^1, a_2^2, a_2^3\}$. The invariants of the locations of $\mathcal{T}$ are all TRUE. Since $\mathcal{T}$ at most a single edge from any location l^j to l^k, all edges can be denoted in the form e_{jk}. The set

of player-1 edges is then $\{e_{01}, e_{02}, e_{20}, e_{10}\}$. The location l^3 has been replicated for ease of drawing in $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$. Observe that $f^{\varepsilon_j}(x \leq 1) = x \leq 1 - \varepsilon_j$ and $f^{\varepsilon_j}(y > 1) = y > 1 - \varepsilon_j$. $\qquad\qquad\square$

The construction of $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$ can be simplified if $\varepsilon_j = 0$ (then we do not need locations of the form l_{e_j}). Given a set of states $\widetilde{S}$ of $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$, let $\mathsf{JStates}(\widetilde{S})$ denote the projection of states to $\mathcal{T}$, defined formally by $\mathsf{JStates}(\widetilde{S}) = \{\langle l, \kappa \rangle \in S \mid \langle l, \widetilde{\kappa} \rangle \in \widetilde{S}$ such that $\kappa(x) = \widetilde{\kappa}(x)$ for all $x \in C\}$, where S is the state space and C the set of clocks of $\mathcal{T}$.

Theorem 6. *Let $\mathcal{T}$ be a timed automaton game, $\varepsilon_r \geq 0$ the response time of player 1, and $\varepsilon_j \geq 0$ the jitter of player 1 actions such that both ε_r and ε_j are rational constants. Then, for any ω-regular location objective $\mathsf{Parity}(\Omega^{\mathcal{T}})$ of $\mathcal{T}$, we have $\mathsf{JStates}\left(\llbracket z = 0 \rrbracket \cap \mathsf{WinTimeDiv}_1^{\mathcal{T}^{\varepsilon_j, \varepsilon_r}}(\mathsf{Parity}(\Omega^{\mathcal{T}^{\varepsilon_j, \varepsilon_r}}))\right) = \mathsf{JRWinTimeDiv}_1^{\mathcal{T}, \varepsilon_j, \varepsilon_r}(\mathsf{Parity}(\Omega^{\mathcal{T}}))$, where $\mathsf{JRWinTimeDiv}_1^{\mathcal{T}, \varepsilon_j, \varepsilon_r}(\Phi)$ is the winning set in the jitter-response semantics, $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$ is the timed automaton with the parity function $\Omega^{\mathcal{T}^{\varepsilon_j, \varepsilon_r}}$ described above, and $\llbracket z = 0 \rrbracket$ is the set of states of $\mathcal{T}^{\varepsilon_j, \varepsilon_r}$ with $\widetilde{\kappa}(z) = 0$.*

Example 3 (Differences between various winning modes). Consider the timed automaton $\mathcal{T}$ in Fig. 1. Let the objective of player 1 be $\square(\neg l^3)$, ie., to avoid l^3. The important part of the automaton is the cycle l^0, l^1. The only way to avoid l^3 in a time divergent run is to cycle in between l^0 and l^1 infinitely often. In additional player 1 may choose to also cycle in between l^0 and l^2, but that does not help (or harm) her. In our analysis, we omit such l^0, l^2 cycles. Let the game start from the location l^0. In a run r, let t_1^j and t_2^j be the times when the a_1^1-th transition and the a_1^2-th transitions respectively are taken for the j-th time. The constraints are $t_1^j - t_1^{j-1} \leq 1$ and $t_2^j - t_2^{j-1} > 1$. If the game cycles infinitely often in between l^0 and l^1 we must also have that for all $j \geq 0$, $t_1^{j+1} \geq t_2^j \geq t_1^j$. we also have that if this condition holds then we can construct an infinite time divergent cycle of l^0, l^1 for some suitable initial clock values. Observe that $t_i^j = t_i^0 + (t_i^1 - t_i^0) + (t_i^2 - t_i^1) + \cdots + (t_i^j - t_i^{j-1})$ for $i \in \{1, 2\}$. We need $t_1^{m+1} - t_2^m = (t_1^{m+1} - t_1^m) + \sum_{j=1}^{m} \left\{(t_1^j - t_1^{j-1}) - (t_2^j - t_2^{j-1})\right\} + (t_1^0 - t_2^0) \geq 0$ for all $m \geq 0$. Rearranging, we get the requirement $\sum_{j=1}^{m} \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. Consider the initial state $\langle l^0, x = y = 0 \rangle$. Let $t_1^0 = 1, t_2^0 = 1.1, t_1^j - t_1^{j-1} = 1, t_2^j - t_2^{j-1} = 1 + 10^{-(j+1)}$. We have $\sum_{j=1}^{m} \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq \sum_{j=1}^{\infty} 10^{-(j+1)} = 10^{-2} * \frac{1}{0.9} \leq 1 - 0.1 = (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. Thus, we have an infinite time divergent trace with the given values. Hence $\langle l^0, x = y = 0 \rangle \in \mathsf{WinTimeDiv}_1^{\mathcal{T}}(\square(\neg l^3))$. It can also be similarly seen that $\langle l^0, x = y = 1 \rangle \in \mathsf{WinTimeDiv}_1(\square(\neg l^3))$ (taking $t_1^0 = 0$ and $t_2^0 = 0.1$).

We now show $\langle l^0, x = y = 0 \rangle \in \mathsf{RobWinTimeDiv}_1(\square(\neg l^3))$. Consider $t_1^0 \in [0.9, 1], t_1^j - t_1^{j-1} \in [1 - 10^{-(j+1)}, 1], t_2^0 \in [1.05, 1.1], t_2^j - t_2^{j-1} \in [1 + 0.5 * 10^{-(j+1)}, 1 + 10^{-(j+1)}]$. We have $\sum_{j=1}^{m} \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} \leq \sum_{j=1}^{m} 10^{-(j+1)} - (-10^{-(j+1)}) \leq 2 * \sum_{j=1}^{\infty} 10^{-(j+1)} = 2 * 10^{-2} * \frac{1}{0.9}$. We also have $(t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0) \geq 1 - 10^{-(m+2)} + (0.9 - 1.1) \geq 0.7$. Thus, we have $\sum_{j=1}^{m} \left\{(t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1})\right\} < 2 * 10^{-2} * \frac{1}{0.9} < 0.7 \leq (t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0)$. This shows that we can construct an infinite cycle in between l^0 and l^1 for all the values in our chosen intervals, and hence that $\langle l^0, x = y = 0 \rangle \in \mathsf{RobWinTimeDiv}_1(\square(\neg l^3))$. Observe that $\langle l^0, x = y = 1 \rangle \notin \mathsf{RobWinTimeDiv}_1(\square(\neg l^3))$

We next show that $\langle l^0, x = y = 0 \rangle \notin \mathsf{JRWinTimeDiv}_1^{\varepsilon_j, \varepsilon_r}(\square(\neg l^3))$ for any $\varepsilon_j > 0$. Observe that for any objective Φ, we have $\mathsf{JRWinTimeDiv}_1^{\varepsilon_j, \varepsilon_r}(\Phi) \subseteq \mathsf{JRWinTimeDiv}_1^{\varepsilon_j, 0}(\Phi)$. Let $\varepsilon_j = \epsilon$ and let $\varepsilon_r = 0$. Consider any player-1 ϵ-jitter 0-response time strategy π_1 that makes the game cycle in between

l^0 and l^1. Player 2 then has a strategy which "jitters" the player-1 moves by ϵ. Thus, the player-1 strategy π_1 can only propose a_1^1 moves with the value of x being less than or equal to $1 - \epsilon$ (else the jitter would make the move invalid). Thus, player 2 can ensure that $t_1^j - t_1^{j-1} \leq 1 - \epsilon$ for all j for some run (since x has the value $t_1^j - t_1^{j-1}$ when a_1^1 is taken for the j-th time for $j > 0$). We then have that for any player-1 ϵ-jitter 0-response time strategy, player 2 has a strategy such that for some resulting run, we have $t_1^j - t_1^{j-1} \leq 1 - \epsilon$ and $t_2^j - t_2^{j-1} > 1$. Thus, $\sum_{j=1}^m \left\{ (t_2^j - t_2^{j-1}) - (t_1^j - t_1^{j-1}) \right\} > m * \epsilon$, which can be made arbitrarily large for a sufficiently large m for any ϵ and hence greater than $(t_1^{m+1} - t_1^m) + (t_1^0 - t_2^0) \leq 1 + (t_1^0 - t_2^0)$ for any initial values of t_1^0 and t_2^0. This violates the requirement for an infinite l^0, l^1 cycle. Thus, $\langle l^0, x = y = 0 \rangle \notin \mathsf{JRWinTimeDiv}_1^{\epsilon,0}(\square(\neg l^3))$ for any $\epsilon > 0$. $\qquad\square$

Theorem 7. *Let $\mathcal{T}$ be a timed automaton and Φ an objective. For all $\varepsilon_{\mathsf{j}} > 0$ and $\varepsilon_{\mathsf{r}} \geq 0$, we have $\mathsf{JRWinTimeDiv}_1^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}(\Phi) \subseteq \mathsf{RobWinTimeDiv}_1(\Phi) \subseteq \mathsf{WinTimeDiv}_1(\Phi)$. All the subset inclusions are strict in general.*

Sampling semantics. Instead of having a response time for actions of player 1, we can have a model where player 1 is only able to take actions in an ε_{j} interval around sampling times, with a given time period $\varepsilon_{\mathsf{sample}}$. A timed automaton can be constructed along similar lines to that of $\mathcal{T}^{\varepsilon_{\mathsf{j}}, \varepsilon_{\mathsf{r}}}$ to obtain the winning set.

References

1. B. Adler, L. de Alfaro, and M. Faella. Average reward timed games. In *FORMATS 05*, LNCS 3829, pages 65–80. Springer, 2005.
2. M. Agrawal and P.S. Thiagarajan. Lazy rectangular hybrid automata. In *HSCC*, LNCS 2993, pages 1–15, 2004.
3. R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
4. R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR 97*, LNCS 1243, pages 74–88. Springer, 1997.
5. R. Alur, S.L. Torre, and P. Madhusudan. Perturbed timed automata. In *HSCC*, LNCS 3414, pages 70–85. Springer, 2005.
6. P. Bouyer, N. Markey, and P.A. Reynier. Robust analysis of timed automata *via* channel machines. In *FoSSaCS 08*, LNCS 4962, pages 157–171. Springer, 2008.
7. F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR 05*, LNCS 3653, pages 66–80. Springer, 2005.
8. K. Chatterjee, T.A. Henzinger, and V.S. Prabhu. Trading infinite memory for uniform randomness in timed games. In *HSCC 08*, LNCS. Springer, 2008.
9. L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 03*, LNCS 2761, pages 144–158. Springer, 2003.
10. D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS 02*, LNCS 2285, pages 571–582. Springer, 2002.
11. M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *VMCAI 02*, LNCS 2294, pages 94–108. Springer, 2002.
12. M. Faella, S. La Torre, and A. Murano. Dense real-time games. In *LICS 02*, pages 167–176. IEEE Computer Society, 2002.
13. V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata. In *HART: Hybrid and Real-Time Systems*, LNCS 1201, pages 331–345. Springer, 1997.
14. T.A. Henzinger and V.S. Prabhu. Timed alternating-time temporal logic. In *FORMATS 06*, LNCS 4202, pages 1–17. Springer, 2006.
15. T.A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *HSCC*, LNCS 1790, pages 145–159. Springer, 2000.
16. M. Jurdzinski. Small progress measures for solving parity games. In *STACS'00*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.

17. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS 95*, pages 229–242, 1995.
18. Sven Schewe. Solving parity games in big steps. In *Proc. FST TCS*. Springer-Verlag, 2007.
19. R. Segala, R. Gawlick, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. *Inf. Comput.*, 141(2):119–171, 1998.
20. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.
21. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV'00*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.
22. H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proc. of 30th Conf. Decision and Control*, pages 1527–1528, 1991.
23. M. D. Wulf, L. Doyen, N. Markey, and J.F. Raskin. Robustness and implementability of timed automata. In *FORMATS/FTRTFT*, LNCS 3253, pages 118–133. Springer, 2004.
24. M. D. Wulf, L. Doyen, and J. F. Raskin. Almost asap semantics: from timed models to timed implementations. *Formal Asp. Comput.*, 17(3):319–341, 2005.

5 Appendix

Representing regions. A region of a timed automaton game $\mathcal{T}$ can be represented as a tuple $R = \langle l, h, \mathcal{P}(C) \rangle$ where (a) l is a location of $\mathcal{T}$; (b) h is a function which specifies the integer values of clocks $h : C \to (\mathbb{N} \cap [0, M])$ (M is the largest constant in $\mathcal{T}$); and (c) $\mathcal{P}(C)$ is a disjoint partition of the clocks $\{C_{-1}, C_0, \ldots C_n \mid \uplus C_i = C, C_i \neq \emptyset$ for $i > 0\}$. Then, a state s with clock valuation κ is in the region corresponding to R when all the following conditions hold: (a) the location of s corresponds to the location of R; (b) for all clocks x with $\kappa(x) \leq c_x$, $\lfloor \kappa(x) \rfloor = h(x)$; (c) for $\kappa(x) > c_x$, $h(x) = c_x$; (d) for all pair of clocks (x, y), with $\kappa(x) \leq c_x$ and $\kappa(y) \leq c_y$, we have $\mathsf{frac}(\kappa(x)) < \mathsf{frac}(\kappa(y))$ iff $x \in C_i$ and $y \in C_j$ with $0 \leq i < j$ (so, $x, y \in C_k$ with $k \geq 0$ implies $\mathsf{frac}(\kappa(x)) = \mathsf{frac}(\kappa(y))$); (e) for $\kappa(x) \leq c_x$, $\mathsf{frac}(\kappa(x)) = 0$ iff $x \in C_0$; and (f) $x \in C_{-1}$ iff $\kappa(x) > c_x$.

Proof of Lemma 3.
Let us denote the region $\widehat{R}$ by $\langle l_1, tick, bl_1, h, \langle C_{-1}, C_0, \ldots, C_n \rangle \rangle$ according to the representation mentioned above. When time elapses, the sets $C_0, \ldots, C_n$ move in a cyclical fashion, i.e., mod $n + 1$. The displacement mod $n + 1$ indicates the relative ordering of the fractional sets. A movement of a "full" cycle of the displacements increases the value of the integral values of all the clocks by 1. We also only track the integral value of a clock $x \in C$ upto c_x, after that the clock is placed into the set C_{-1}. Note that the extra clock z introduced in $\widehat{\mathcal{T}}$ is never placed into C_{-1}, and always has a value mod 1. Let us order the clocks in C in order of their increasing c_x values, i.e., $c_{x_1} \leq c_{x_2} \leq \ldots c_{x_N}$ where $N = C$. The most number of time successors are obtained when all clocks have an integral value of 0 to start with. We count the number of time successors in N stages. In the first stage, $C_{-1} = \emptyset$. After at most c_{x_1} full cycles, the clock x_1 gets moved to C_{-1} as its value exceeds the maximum tracked value. For each full cycle, we also have the number of distinct mod classes to be $N + 1$ (recall that we also have the extra clock z). We need another factor of 2 to account for the movement which makes all clock values non-integral, e.g., $\langle x = 1, y = 1.2, z = 0.99 \rangle$ to $\langle x = 1.00001, y = 1.20001, z = 0.99001 \rangle$. Thus, before the clock c_{x_1} gets moved to C_{-1}, we can have $2 \cdot (c_{x_1} + 1) \cdot (N + 1)$ time successors. In the second stage, we can have at most $c_{x_2} + 1 - c_{x_1}$ before clock c_{x_2} gets placed into C_{-1}. Also, since x_1 is in C_{-1}, we can only have $N + 1 - 1$ mod classes in the second stage. Thus, the number of time successors added in the second stage is at most $2 \cdot (c_{x_2} + 1 - c_{x_1}) \cdot N$. Continuing in this fashion, we obtain the total number of time successors as $2 \cdot$

$$\left((c_{x_1} + 1) \cdot (N+1) + (c_{x_2} + 1 - c_{x_1}) \cdot (N+1-1) + \cdots + (c_{x_N} + 1 - \textstyle\sum_{i=1}^{N-1} c_{x_i}) \cdot (N+1-(N-1)) \right)$$
$$= 4 \cdot \textstyle\sum_{i=1}^{N} (c_{x_i} + 1). \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \square$$

Construction of the finite turn based game $\mathcal{T}^f$.
The game $\mathcal{T}^f$ consists of a tuple $\langle S^f, E^f, S_1^f, S_2^f \rangle$ where,

- $S^f = S_1^f \cup S_2^f$ is the state space. The states in S_i^f are controlled by player-i for $i \in \{1,2\}$.
- $S_1^f = \widehat{S}_{\mathsf{Reg}} \times \{1\}$, where $\widehat{S}_{\mathsf{Reg}}$ is the set of regions in $\widehat{\mathcal{T}}$.
- $S_2^f = \widehat{S}_{\mathsf{Tup}} \times \{2\}$.
  The set $\widehat{S}_{\mathsf{Tup}}$ will be described later. Intuitively, a $B \in \widehat{S}_{\mathsf{Tup}}$ represents a 3-tuple $\langle Y_1, Y_2, a_1 \rangle$ where Y_i are regions of $\widehat{\mathcal{T}}$, such that $\langle \Delta, a_1 \rangle \in \widehat{\Gamma}_1(\widehat{s})$ with $\widehat{s} + \Delta \in Y_2$. The values of Y_2 and a_1 are maintained indirectly.
- $\widehat{S}_{\mathsf{Tup}} = L \times \{\mathrm{TRUE}, \mathrm{FALSE}\}^2 \times H \times \mathcal{P}(\widehat{C}) \times \{0, \ldots, M\} \times \{0, \ldots, |C|+1\} \times \{\mathrm{TRUE}, \mathrm{FALSE}\} \times L \times 2^C \times \{\mathrm{TRUE}, \mathrm{FALSE}\}$, where H is the set of valuations from C to positive integers such that each clock x is mapped to a value less than or equal to c_x where c_x is the largest constant to which clock x is compared to.
  Given $Z = \langle l_1, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, k, w, om, l_2, \lambda, tev \rangle \in \widehat{S}_{\mathsf{Tup}}$, we let $\mathsf{FirstRegion}(Z)$ denote the region $\langle l_2, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle \rangle \in \widehat{S}_{\mathsf{Reg}}$. Intuitively, $\mathsf{FirstRegion}(Z)$ is the region from which player 1 first proposes a move. The move of player 1 consists of a intermediate region Y, denoting that first time passes to let state change from $\mathsf{FirstRegion}(Z)$ to Y; and a discrete jump action specified by a destination location l_2, together with the clocks to be reset, λ (we observe that the discrete actions may also be directly specified as $a_1 \in A_1$ in case $|A_1| \leq |L| \cdot 2^C$). The variable tev is true iff player 1 proposed any relinquishing time move. The region Y is obtained from Z using the variables $0 \leq k \leq M$, $0 \leq w \leq |C|+1$, and $om \in \{\mathrm{TRUE}, \mathrm{FALSE}\}$. The integer w indicates the the relative movement of the clock fractional parts $C_0, \ldots C_n$ (note that the movement must occur in a cyclical fashion). The integer k indicates the number of cycles completed. It can be at most M because after that, all clock values become bigger that the maximum constant, and thus need not be tracked. The boolean variable om indicates whether a small ϵ-move has taken place so that no clock value is integral, eg., $\langle x=1, y=1.2, z=0.99 \rangle$ to $\langle x=1.00001, y=1.20001, z=0.99001 \rangle$.
  Formally, $\mathsf{SecondRegion}(Z)$ denotes the region $\langle l_2, tick', bl_1, h', \langle C'_{-1}, \ldots, C'_m \rangle \rangle \in \widehat{S}_{\mathsf{Reg}}$ where

  - $h'(x) = \begin{cases} h(x) + k & \text{if } h(x)+k \leq c_x \text{ and } x \in C_j \text{ with } j+w \leq n; \\ h(x) + k + 1 & \text{if } h(x)+k+1 \leq c_x \text{ and } x \in C_j \text{ with } j+w > n; \\ c_x & \text{otherwise.} \end{cases}$
    The integer k indicates the number of integer boundaries crossed by all the clocks when getting to the new region. Some clocks may cross k integer boundaries, while others may cross $k+1$ integer boundaries.
  - $h_{\max}(x) = \begin{cases} h(x) + k & \text{if } x \in C_j \text{ with } j+w \leq n; \\ h(x) + k + 1 & \text{if } x \in C_j \text{ with } j+w > n. \end{cases}$
    ($h_{\max}$ will be used later in the definition of $f_{\max}^{h_{\max}}$.)
  - $\langle C'_{-1}, \ldots, C'_m \rangle = f_{\mathsf{Compact}} \circ f_{\max}^{h_{\max}} \circ f_{\mathsf{OpenMove}}^{om} \circ f_{\mathsf{Cycle}}^w (\langle C_{-1}, \ldots, C_n \rangle)$, where
    * $f_{\mathsf{Cycle}}^w (\langle C_{-1}, \ldots, C_n \rangle) = \langle C_{-1}, C'_0, \ldots, C'_n \rangle$ with $C'_{(j+w) \bmod (n+1)} = C_j$.
      This function cycles around the fractional parts by w.

* $f^{om}_{\mathsf{OpenMove}}(\langle C_{-1}, C_0, \dots, C_n\rangle) = \begin{cases} \langle C_{-1}, C_0, \dots, C_n\rangle & \text{if } om = \text{FALSE}; \\ \langle C_{-1}, \emptyset, C_0, \dots, C_n\rangle & \text{if } om = \text{TRUE}. \end{cases}$

  This function indicates if the current region is such that all the clocks have non-integral values (if $om = \text{TRUE}$).

* $f_{\max}(\langle C_{-1}, C_0, \dots, C_n\rangle) = \langle C'_{-1}, C'_0, \dots, C'_n\rangle$ with $C'_j = C_j \setminus V_j$ for $j \geq 0$ and $C'_{-1} = C_{-1} \cup_{j=0}^{n} V_j$ where (a) $x \in V_0$ iff $x \in C_0$ and $h_{\max}(x) > c_x$; and (b) $x \in V_j$ for $j > 0$ iff $x \in C_j$ and $h'(x) = c_x$.

  When clocks are cycled around, some of them may exceed the maximal tracked values c_x. In that case, they need to be moved to C_{-1}. This function is accomplished by $f_{\max}$.

* $f_{\mathsf{Compact}}(\langle C_{-1}, C_0, \dots, C_m\rangle)$ eliminates the empty sets for $j > 0$. It can be obtained by the following procedure:

  $i := 0, j := 1$
  while $j \leq m$ **do**
     while $j < m$ and $C_j = \emptyset$ **do**
        $j := j + 1$
     end while
     if $C_j \neq \emptyset$ **then**
        $C_{i+1} := C_j$
        $i := i + 1, j := j + 1$
     end if
  end while
  return $\langle C_{-1}, C_0, \dots, C_i\rangle$

- $tick' = \text{TRUE}$ iff $k > 0$; or $z \in C_i$ and $w > n - i$.

– The set of edges is specified by a transition relation δ^f, and a set of available moves Γ^f_i. We let A^f_i denote the set of moves for player-i, and $\Gamma_i(X)$ denote the set of moves available to player-i at state $X \in S^f_i$.

– $A^f_1 = (\widehat{S}_{\mathsf{Reg}} \times L \times 2^C \cup \{\perp_1\}) \times \{1\}$.

  The component $\widehat{S}_{\mathsf{Reg}}$ denotes the region that player 1 wants to let time elapse to in $\widehat{\mathcal{T}}$ to before she takes a jump with the destination specified by the location and the set of clocks that are reset. The move $\{\perp_1\} \times \{1\}$ is a relinquishing move, corresponding to a pure time move in $\widehat{\mathcal{T}}$.

– $A^f_2 = \widehat{S}_{\mathsf{Reg}} \times \{1, 2\} \times L \times 2^C \times \{2\}$.

  The component $\widehat{S}_{\mathsf{Reg}}$ denotes the region that player 2 wants to let time elapse to in $\widehat{\mathcal{T}}$ to before she takes a jump with the destination specified by the location and the set of clocks that are reset. The element in $\{1, 2\}$ is used in the case player 2 picks the same intermediate region $\widehat{S}_{\mathsf{Reg}}$ as player 1. In this case, player 2 has a choice of letting the move of player 1 win or not, and the number from $\{1, 2\}$ indicates which player wins.

– The set of available moves for player 1 at a state $\langle X, 1\rangle$ is given by $\Gamma^f_1(X \times \{1\}) = \{\perp_1\} \times \{1\} \cup$
$$\left\{ \langle Y, l_y, \lambda, 1\rangle \;\middle|\; \begin{array}{l} \exists \widehat{s} = \langle l_x, \widehat{\kappa}_x\rangle \in X, \exists \langle \Delta, \perp\rangle \in \widehat{\Gamma}_1(\widehat{s}) \text{ such that } \langle l_x, \widehat{\kappa}_x\rangle + \Delta \in Y \text{ and} \\ \exists \widehat{s}' \in Y, \exists \langle l_x, a_1, \theta, l_y, \lambda\rangle \in \widehat{\Gamma}_1(\widehat{s}'), \text{ such that } \widehat{s}' \models \theta \end{array} \right\}$$

– The set of available moves for player 2 at a state $\langle X, 2\rangle$ is given by $\Gamma^f_1(X \times \{2\}) =$
$$\left\{ \langle Y, i, l_y, \lambda, 2\rangle \;\middle|\; \begin{array}{l} i \in \{1, 2\}, \exists \widehat{s} = \langle l_x, \widehat{\kappa}_x\rangle \in \mathsf{FirstRegion}(\langle X, 2\rangle), \exists \langle \Delta, \perp_2\rangle \in \widehat{\Gamma}_2(\widehat{s}) \\ \text{such that } \langle l_x, \widehat{\kappa}_x\rangle + \Delta \in Y \text{ and} \\ \text{(a) } l_y = l_x \text{ and } \lambda = \emptyset \text{ or,} \\ \text{(b) } \exists \widehat{s}' \in Y \; \exists \langle l_x, a_2, \theta, l_y, \lambda\rangle \in \Gamma_2(\widehat{s}') \text{ such that } \widehat{s}' \models \theta \end{array} \right\}$$

– The transition function δ^f is specified by
  - $\delta^f(\langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, 1 \rangle, \langle Y, l_y, \lambda, 1 \rangle) =$
    $\langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, k, w, om, l_y, \lambda, \text{FALSE}, 2 \rangle$, where $0 \leq k \leq M, 0 \leq w \leq |C| + 1, om \in \{\text{TRUE}, \text{FALSE}\}$ are such that $Y = \text{SecondRegion}(\langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, k, w, om, l_y, \lambda, \text{FALSE}, 2 \rangle)$.
  - $\delta^f(\langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, 1 \rangle, \langle \bot, 1 \rangle) =$
    $\langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, 0, 0, \text{FALSE}, l, \emptyset, \text{TRUE}, 2 \rangle$.
  - Let $\langle Z, 2 \rangle = \langle l, tick, bl_1, h, \langle C_{-1}, \ldots, C_n \rangle, k, w, om, l_z, \lambda_z, tev, 2 \rangle)$.
    Then, $\delta^f(\langle Z, 2 \rangle, \langle Y, 2, l_y, \lambda_y, 2 \rangle) =$

$$\begin{cases} \langle \text{SecondRegion}(Z)[\text{loc} := l_z, \lambda_z := 0, bl_1 := \text{TRUE}], 1 \rangle & \text{if } tev = \text{FALSE and all player 1} \\ & \text{moves to SecondRegion}(Z) \\ & \text{beats all player 2 moves to } Y \\ & \text{from the region FirstRegion}(Z) \\ & \text{according to Lemma 2;} \\ \langle Y[loc := l_y, \lambda_y := 0, bl_1 = \text{FALSE}], 1 \rangle & \text{otherwise.} \end{cases}$$

  - $\delta^f(\langle Z, 2 \rangle, \langle Y, 1, l_y, \lambda_y, 2 \rangle) =$

$$\begin{cases} \langle \text{SecondRegion}(Z)[\text{loc} := l_z, \lambda_z := 0, bl_1 = \text{TRUE}], 1 \rangle & \text{if } tev = \text{FALSE and all player 1} \\ & \text{moves to SecondRegion}(Z) \\ & \text{beats all player 2 moves to } Y \\ & \text{from the region FirstRegion}(Z) \\ & \text{according to Lemma 2;} \\ \langle \text{SecondRegion}(Z)[\text{loc} := l_z, \lambda_z := 0, bl_1 := \text{TRUE}], 1 \rangle & \text{if } tev = \text{FALSE and} \\ & Y = \text{SecondRegion}(Z) \text{ ie., both} \\ & \text{players pick the same time delay,} \\ & \text{(and player 2 allows the player 1 move,} \\ & \text{signified by the 1 in} \langle Y, 1, l_y, \lambda_y, 2 \rangle); \\ \langle Y[loc := l_y, \lambda_y := 0, bl_1 := \text{FALSE}], 1 \rangle & \text{otherwise..} \end{cases}$$

Note that we change the values of bl_1 and $tick$ only after player-2 moves.

Complexity of reduction.
Let $|A_i|^* = \min\{|A_i|, L \cdot 2^{|C|}\}$ for $i \in \{1, 2\}$. In the construction of $\mathcal{T}^f$, we can keep track of actions, or the locations together with the reset sets depending on whether $|A_i|$ is bigger than $L \cdot 2^{|C|}$ or not. We have $|S_1^f| = |\widehat{S}_{\text{Reg}}|$, and $|S_2^f| = |\widehat{S}_{\text{Reg}}| \cdot (M + 1) \cdot (|C| + 2) \cdot 2 \cdot (|A_1|^* + 1)$ (we have incorporated a modification where we represent possible actions by $\{\bot_1\} \cup A_1$ instead of $L \times 2^C \times \{\text{TRUE}, \text{FALSE}\}$). Given a state $Z \in S_1^f$, the number player-1 edges from Z is equal to one plus the cardinality of the set of time successors of Z multiplied by player-1 actions. This is equal to $(|A_1| + 1)^* \cdot ((M + 1) \cdot (|C| + 2) \cdot 2)$ (the $+1$ corresponds to the relinquishing move). Thus the total number of player-1 edges is at most $|\widehat{S}_{\text{Reg}}| \cdot (|A_1|^* + 1) \cdot ((M + 1) \cdot (|C| + 2) \cdot 2)$. Given a state $X \in S_2^f$, the number player-2 edges from X is equal to $2 \cdot (|A_2|^* + 1)$ multiplied by the cardinality of the set of time successors of $\text{FirstRegion}(X)$ (the plus one arises as player-2 can have a pure time move in addition to actions from A_2). Thus, the number of player-2 edges is at most $|S_2^f| \cdot 2 \cdot (|A_2|^* + 1) \cdot ((M + 1) \cdot (|C| + 2) \cdot 2)$. Hence, $|E^f| \leq |\widehat{S}_{\text{Reg}}| \cdot ((M + 1) \cdot (|C| + 2) \cdot 2) \cdot (|A_1|^* + 1) [(1 + (|A_2|^* + 1) \cdot ((M + 1) \cdot (|C| + 2) \cdot 2)]$. Let $|\mathcal{T}_{\text{Constr}}|$ denote the length of the clock constraints in $\mathcal{T}$. For our complexity analysis, we assume all clock constraints are in conjunctive normal form. For constructing $\mathcal{T}^f$, we need to check whether regions satisfy clock constraints from $\mathcal{T}$. For this,

we build a list of regions with valid invariants together with edge constraints satisfied at the region. This takes $O(|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|)$ time. We assume a region can be represented in constant space.

Proof of Theorem 3

. From [18], we have that a turn based parity game with m edges, n states and d parity indices can be solved in $O(m \cdot n^{\frac{d}{3}+\frac{1}{2}})$ time. Thus, $\mathsf{WinTimeDiv}_1^{\mathcal{T}}(\mathsf{Parity}(\Omega))$ can be computed in time $O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + \mathcal{F}_1 \cdot \mathcal{F}_2^{\frac{d+2}{3}+\frac{1}{2}}\right)$, where $\mathcal{F}_1 = |\widehat{S}_{\mathsf{Reg}}| \cdot ((M+1) \cdot (|C|+2) \cdot 2) \cdot (|A_1|^* + 1) \left[(1 + (|A_2|^* + 1) \cdot ((M+1) \cdot (|C|+2) \cdot 2)\right]$, and $\mathcal{F}_2 = |\widehat{S}_{\mathsf{Reg}}| \cdot (1 + (M+1) \cdot (|C|+2) \cdot 2 \cdot (|A_1|^* + 1))$, which is equal to

$$O\left((|\widehat{S}_{\mathsf{Reg}}| \cdot |\mathcal{T}_{\mathsf{Constr}}|) + [M \cdot |C| \cdot |A_2|^*] \cdot \left[2 \cdot |\widehat{S}_{\mathsf{Reg}}| \cdot M \cdot |C| \cdot |A_1|^*\right]^{\frac{d+2}{3}+\frac{3}{2}}\right)$$

$\square$

We now present an extension for Lemma 1.

Lemma 4. *Let $\mathcal{T}$ be a timed automaton game and $\widehat{\mathcal{T}}$ be the corresponding enlarged game structure. Let $\widehat{\Phi}$ be an ω-regular region objective of $\widehat{\mathcal{T}}$. If π_1 is a region strategy that is winning for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$ and π_1^{rob} is a robust strategy that is region-equivalent to π_1, then π_1^{rob} is a winning strategy for $\widehat{\Phi}$ from $\mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$.*

Proof. Consider any strategy π_2 for player 2, and a state $\widehat{s} \in \mathsf{Win}_1^{\widehat{\mathcal{T}}}(\widehat{\Phi})$. We have $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2)$ to be the set of runs r such that for all $k \geq 0$, either a) $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta, \perp_1 \rangle$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], \langle \Delta, \perp_1 \rangle, \pi_2(r[0..k]))$ or, $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ and $r[k+1] = \widehat{\delta}_{\mathsf{jd}}(r[k], \langle \Delta, a_1 \rangle, \pi_2(r[0..k]))$ for some $\Delta \in [\alpha, \beta]$. It can be observed that $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) = \bigcup_{\pi_1'} \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$ where π_1' ranges over (non-robust) player-1 strategies such that for runs $r \in \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$ and for all $k \geq 0$ we have $\pi_1'(r[0..k]) = \langle \Delta, \perp_1 \rangle$ if $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle \Delta, \perp_1 \rangle$, and $\pi_1'(r[0..k]) = \langle \Delta, a_1 \rangle$ if $\pi_1^{\mathsf{rob}}(r[0..k]) = \langle [\alpha, \beta], a_1 \rangle$ for some $\Delta \in [\alpha, \beta]$; and π_1' acts like π_1 otherwise (note that the runs r and the strategies π_1' are defined inductively with respect to k, with $r[0] = \widehat{s}$). Each player-1 strategy π_1' in the preceeding union is region equivalent to π_1 since π_1^{rob} is region equivalent to π_1 and hence each π_1' is a winning strategy for player 1 by Lemma 1. Thus, $\mathsf{Outcomes}(s, \pi_1^{\mathsf{rob}}, \pi_2) = \bigcup_{\pi_1'} \mathsf{Outcomes}(\widehat{s}, \pi_1', \pi_2)$ is a subset of $\widehat{\Phi}$, and hence π_1^{rob} is a winning strategy for player 1. $\square$